

[MS-FSFDMW]: FAST Distributed Make Worker Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.msp>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplq@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
02/19/2010	1.0	Major	Initial Availability
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1	Introduction	9
1.1	Glossary	9
1.2	References.....	10
1.2.1	Normative References.....	10
1.2.2	Informative References	10
1.3	Protocol Overview (Synopsis)	11
1.4	Relationship to Other Protocols.....	11
1.5	Prerequisites/Preconditions	12
1.6	Applicability Statement.....	12
1.7	Versioning and Capability Negotiation.....	12
1.8	Vendor-Extensible Fields.....	12
1.9	Standards Assignments	12
2	Messages.....	13
2.1	Transport.....	13
2.2	Message Syntax	13
2.2.1	Simple Types	14
2.2.1.1	Task Identifiers	14
2.2.1.2	Path Names	14
2.2.1.3	Transfer Key	15
2.2.2	Enumerations	16
2.2.2.1	Task Type Enumeration	16
2.2.2.2	Task Status Enumeration	16
2.2.2.3	Merge Function Enumerations	16
2.2.2.4	Pythontasks Functions Enumerations.....	17
2.2.2.5	Reduce Functions Enumerations	18
2.2.2.6	Shell Command Enumerations.....	18
2.2.3	Complex Types	19
2.2.3.1	Dir_size Tuple Structure	19
2.2.3.2	Heartbeat_info Structure	19
2.2.3.3	Path_dir_size Tuple Structure.....	20
2.2.3.4	Directory_listing Array	20
2.2.3.5	File_listing Array.....	20
2.2.3.6	Task_info Structure	20
2.2.3.7	Task_info_copy Structure	21
2.2.3.8	Sortkeyspec String	21
2.2.3.9	Tableinputspec String.....	22
2.2.3.10	Tableoutputspec String	22
2.2.3.11	Target_name String	22
2.2.4	Abort Command.....	23
2.2.5	CD Command	23
2.2.6	Close-stdin Command	23
2.2.7	Disk_limit Command.....	23
2.2.8	Error Message.....	23
2.2.9	Finish Message.....	24
2.2.10	Finish-tasks Command	25
2.2.11	Finish-tasks-ack Message	25
2.2.12	Forget Command.....	25
2.2.13	Heartbeat Message	25
2.2.14	Info Message	25

2.2.15	Netsink Command	25
2.2.16	Netsource Command	26
2.2.17	Poll-task Command	26
2.2.18	Remove-files Command	26
2.2.19	Request_heartbeat Command	27
2.2.20	Setenv Command	27
2.2.21	Shell Command	27
2.2.21.1	Cobra Command	28
2.2.21.1.1	Merge.pyc Command	28
2.2.21.1.2	Pythontasks.pyc Command	28
2.2.21.1.2.1	pythontasks functions and parameters	29
2.2.21.1.3	Reduce.pyc Command	30
2.2.21.1.4	Split.pyc Command	30
2.2.21.2	Other Shell Commands	31
2.2.21.2.1	Create_attribute_files Command	31
2.2.21.2.2	Jsort2.exe Command	31
2.2.21.2.3	Make_pu_diff Command	32
2.2.21.2.4	Pupdateclient Command	32
2.2.21.2.5	Sp_make_pu_diff Command	32
2.2.21.2.6	Truncate_anchorinfo Command	33
2.2.21.2.7	Truncate_clickinfo Command	33
2.2.22	Simple_command Command	34
2.2.23	Simple_response Message	35
2.2.24	Slot Message	35
2.2.25	Stderr Message	36
2.2.26	Stdout Message	36
2.2.27	Task-info Message	37

3	Protocol Details	38
3.1	Common Details	38
3.1.1	Abstract Data Model	38
3.1.2	Timers	38
3.1.3	Initialization	38
3.1.4	Higher-Layer Triggered Events	39
3.1.5	Message Processing Events and Sequencing Rules	39
3.1.6	Timer Events	39
3.1.7	Other Local Events	39
3.2	Master Protocol Server Details	39
3.2.1	Abstract Data Model	39
3.2.2	Timers	39
3.2.2.1	Heartbeat Timeout Timer	39
3.2.2.2	Heartbeat Timer	39
3.2.2.3	Build Timeout Timer	39
3.2.3	Initialization	40
3.2.4	Higher-Layer Triggered Events	40
3.2.4.1	Starting Build	40
3.2.4.2	Starting Command on Worker Protocol Client	40
3.2.4.3	Sending Remove-files Command	40
3.2.4.4	Sending Shell Command	41
3.2.4.5	Sending Abort Command	41
3.2.4.6	Sending Finish-tasks Message	41
3.2.4.7	Sending Setenv Command	41
3.2.5	Message Processing Events and Sequencing Rules	41

3.2.5.1	Info Message	41
3.2.5.2	Finish Message	41
3.2.5.3	Finish-tasks-ack Message	41
3.2.5.4	Heartbeat Message	41
3.2.5.5	Netsink-Netsource Transaction	42
3.2.5.6	Simple_response Message	42
3.2.5.7	Stderr Message	42
3.2.5.8	Stdout Message	42
3.2.5.9	Task-info Message	43
3.2.6	Timer Events	43
3.2.6.1	Heartbeat Timeout	43
3.2.7	Other Local Events	43
3.3	Protocol Client Worker Details	43
3.3.1	Abstract Data Model	43
3.3.2	Timers	43
3.3.3	Initialization	43
3.3.4	Higher-Layer Triggered Events	44
3.3.4.1	Stable Storage Capacity Shortage	44
3.3.5	Message Processing Events and Sequencing Rules	44
3.3.5.1	Generic-response Messages	44
3.3.5.1.1	Remove-files Message	45
3.3.5.1.2	Shell Message	45
3.3.5.1.2.1	Table Functions	46
3.3.5.2	Specific-response Messages	47
3.3.5.2.1	Receiving Abort Message	47
3.3.5.2.2	Receiving Forget Message	47
3.3.5.2.3	Receiving Poll-task Message	47
3.3.5.2.4	Receiving Request_heartbeat Message	47
3.3.5.2.5	Receiving Simple_command Message	48
3.3.5.2.5.1	create_builddirs	48
3.3.5.2.5.2	create_builddirs_no_tmp	48
3.3.5.2.5.3	find_builddirs	49
3.3.5.2.5.4	link_to_file	49
3.3.5.2.5.5	markfile	49
3.3.5.2.5.6	remove	49
3.3.5.2.5.7	touch	49
3.3.5.3	No-response Messages	49
3.3.5.3.1	Receiving CD Message	50
3.3.5.3.2	Receiving Close-stdin Message	50
3.3.5.3.3	Receiving Disk_limit Message	50
3.3.5.3.4	Receiving Finish-tasks Message	50
3.3.5.3.5	Receiving Info Message	50
3.3.5.3.6	Receiving Setenv Message	50
3.3.6	Timer Events	50
3.3.7	Other Local Events	50
3.4	Netsink Client Details	50
3.4.1	Abstract Data Model	50
3.4.2	Timers	51
3.4.3	Initialization	51
3.4.4	Higher-Layer Triggered Events	51
3.4.5	Message Processing Events and Sequencing Rules	51
3.4.6	Timer Events	53
3.4.7	Other Local Events	53

3.5	Netsource Client Details.....	53
3.5.1	Abstract Data Model	53
3.5.2	Timers	54
3.5.3	Initialization	54
3.5.4	Higher-Layer Triggered Events.....	54
3.5.5	Message Processing Events and Sequencing Rules.....	54
3.5.6	Timer Events	55
3.5.7	Other Local Events	55
3.6	Child Process Shell Cobra Client.....	55
3.6.1	Abstract Data Model	55
3.6.2	Timers	55
3.6.3	Initialization	55
3.6.4	Higher-Layer Triggered Events.....	55
3.6.5	Message Processing Events and Sequencing Rules.....	55
3.6.5.1	Receiving Merge.pyc Command	56
3.6.5.1.1	Receiving Spmakefileutils.merge_with_global_count Message.....	56
3.6.5.1.2	Receiving Spmakefileutils.merge_with_query Message.....	57
3.6.5.1.3	Receiving Spmakefileutils.merge_with_url Message	57
3.6.5.1.4	Receiving Spmakefileutils.normalize_url Message	58
3.6.5.1.5	Receiving Wamakefileutils.compute_linkscore_divide_by_linkcount Message.....	58
3.6.5.1.6	Receiving Wamakefileutils.compact_links_and_merge_fromsite Message ...	59
3.6.5.1.7	Receiving Wamakefileutils.cut_fromuris_and_reverse_urihash Message	59
3.6.5.1.8	Receiving Wamakefileutils.extend_link_with_freqs Message.....	60
3.6.5.1.9	Receiving Wamakefileutils.filter_intra_and_merge_tosite Message	60
3.6.5.1.10	Receiving Wamakefileutils.logcompact_eqrepr Message.....	61
3.6.5.1.11	Receiving Wamakefileutils.filter_sitemap Message	61
3.6.5.1.12	Receiving Wamakefileutils.logcompact_links Message	62
3.6.5.1.13	Receiving Wamakefileutils.logcompact_no_links Message	63
3.6.5.1.14	Receiving Wamakefileutils.logcompact_urieq Message.....	63
3.6.5.1.15	Receiving Wamakefileutils.make_new_urihashmap Message	64
3.6.5.1.16	Receiving Wamakefileutils.merge_in_rank Message	64
3.6.5.1.17	Receiving Wamakefileutils.merge_repr Message	65
3.6.5.1.18	Receiving Wamakefileutils.merge_siterank Message	65
3.6.5.1.19	Receiving Wamakefileutils.normalize_link Message	66
3.6.5.1.20	Receiving Wamakefileutils.parse_site Message	66
3.6.5.1.21	Receiving Wamakefileutils.reduce_anchor_and_add_rank Message	67
3.6.5.1.22	Receiving Wamakefileutils.reverse_urihash Message	68
3.6.5.1.23	Receiving Wamakefileutils.sum_linkscore_and_new_static_rank Message.....	68
3.6.5.2	Receiving pythontasks.pyc Command.....	69
3.6.5.2.1	Receiving _local_copy_func Message	69
3.6.5.2.2	Receiving _local_touch_func Message	69
3.6.5.2.3	Receiving Spmakefileutils.file_finder Message	69
3.6.5.2.4	Receiving Wamakefileutils.clean_view_build_dir Message	70
3.6.5.2.5	Receiving Wamakefileutils.compare_links_and_nolinks Message	70
3.6.5.2.6	Receiving Wamakefileutils.file_finder Message	70
3.6.5.2.7	Receiving Wamakefileutils.remove_files_in_filter Message	70
3.6.5.2.8	Receiving Wamakefileutils.remove_files_not_in_filter Message	71
3.6.5.3	Receiving Reduce.pyc Command	71
3.6.5.3.1	Receiving Spmakefileutils.compact_uris Message	71
3.6.5.3.2	Receiving Spmakefileutils.local_count_queries Message	72
3.6.5.3.3	Receiving Spmakefileutils.filter_old_queries Message.....	72

3.6.5.3.4	Receiving Spmakefileutils.filter_old_urls Message.....	73
3.6.5.3.5	Receiving Spmakefileutils.full_global_count_queries Message	74
3.6.5.3.6	Receiving Spmakefileutils.full_local_count_queries Message	74
3.6.5.3.7	Receiving Spmakefileutils.logcompact_contentids Message	74
3.6.5.3.8	Receiving Spmakefileutils.make_uris_uniq Message.....	75
3.6.5.3.9	Receiving Spmakefileutils.reduce_localcount Message.....	75
3.6.5.3.10	Receiving Spmakefileutils.reduce_urls Message	76
3.6.5.3.11	Receiving Spmakefileutils.tokenize_query Message.....	76
3.6.5.3.12	Receiving Wamakefileutils.anchor_weight Message.....	77
3.6.5.3.13	Receiving Wamakefileutils.compact_eqrepr Message.....	77
3.6.5.3.14	Receiving Wamakefileutils.compact_links Message	78
3.6.5.3.15	Receiving Wamakefileutils.compact_urieq Message.....	78
3.6.5.3.16	Receiving Wamakefileutils.cut_linkuris Message	78
3.6.5.3.17	Receiving Wamakefileutils.cut_urieq Message	79
3.6.5.3.18	Receiving Wamakefileutils.do_siterank Message.....	79
3.6.5.3.19	Receiving Wamakefileutils.eqrepr_by_urihash_prep Message	79
3.6.5.3.20	Receiving Wamakefileutils.extract_links Message	80
3.6.5.3.21	Receiving Wamakefileutils.initial_rank Message	80
3.6.5.3.22	Receiving Wamakefileutils.links_by_from_prep Message.....	80
3.6.5.3.23	Receiving Wamakefileutils.make_unique Message.....	81
3.6.5.3.24	Receiving Wamakefileutils.reduce_to Message	81
3.6.5.3.25	Receiving Wamakefileutils.urieq_by_class_prep Message.....	82
3.6.5.4	Receiving Split.pyc Command	82
3.6.6	Timer Events	83
3.6.7	Other Local Events	83
3.7	Child Process Shell Client.....	83
3.7.1	Abstract Data Model	83
3.7.2	Timers	83
3.7.3	Initialization	84
3.7.4	Higher-Layer Triggered Events.....	84
3.7.5	Message Processing Events and Sequencing Rules.....	84
3.7.5.1	Create_attribute_files Command	84
3.7.5.2	Jsort2.exe Command	84
3.7.5.3	Make_pu_diff Command	84
3.7.5.4	Pupdateclient Command	85
3.7.5.5	Sp_make_pu_diff Command	85
3.7.5.6	Truncate_anchorinfo Command	85
3.7.5.7	Truncate_clickinfo Command	86
3.7.6	Timer Events	86
3.7.7	Other Local Events	86
4	Protocol Examples.....	87
4.1	Path Name References	87
4.2	cd Message	87
4.3	simple_command Message.....	87
4.4	shell_task	88
4.5	File transfer with netsink and netsource.....	88
5	Security.....	90
5.1	Security Considerations for Implementers.....	90
5.2	Index of Security Parameters	90
6	Appendix A: Product Behavior.....	91

7 Change Tracking.....92

8 Index93

1 Introduction

This document specifies the FAST Distributed Make Worker Protocol (FDMW). Computers that use this protocol perform parallel distributed computing, and the protocol enables them to exchange data and request processing on other computers.

The protocol servers that use this protocol form a **cluster**, one as the master server with a set of worker servers. This protocol is not a traditional client-server protocol, because all computers can perform the role of protocol client and protocol server in different contexts. Within the master server-worker server relationship however, the master server is the protocol server and the worker server is the protocol client.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

Augmented Backus-Naur Form (ABNF)

base64

cluster

Domain Name System (DNS)

environment variables

FileId

process identifier (PID)

UTF-8

The following terms are defined in [\[MS-OFCGLOS\]](#):

base port

content collection

hyperlink

item

ranking

search clickthrough

The following terms are specific to this document:

netsink: A protocol client that is temporarily designated to receive information directly from another protocol client and to perform operations on that information.

netsource: A protocol client that is temporarily authorized to act as a protocol server and send information directly to another protocol client.

pseudo-random number generator (PRNG): An algorithm that is used to generate a sequence of numbers that approximates properties of random numbers. The sequence is not truly random because it is determined by an arbitrary set of initial values.

transfer channel: A packet, datagram, octet stream connection, or sequence of connections that exists directly between two protocol clients, a netsink and a netsource.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[IEEE802.3] Institute of Electrical and Electronics Engineers, "Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Description", IEEE Std 802.3, 2002, <http://standards.ieee.org/getieee802/download/802.3-2002.pdf>

[ISO-9899] International Organization for Standardization, "Programming Languages - C", ISO/IEC 9899:TC2, May 2005, <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf>

[MC-RegEx] Microsoft Corporation, "Regular Expression Language Elements", [http://msdn.microsoft.com/en-us/library/az24scfc\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/az24scfc(VS.80).aspx)

[MS-FSCF] Microsoft Corporation, "[Content Feeding Protocol Specification](#)"

[MS-FSCX] Microsoft Corporation, "[Configuration \(XML-RPC\) Protocol Specification](#)"

[MS-FSIN] Microsoft Corporation, "[Input Normalization Data Structure](#)"

[MS-FSSPRDF] Microsoft Corporation, "[SPRel Data File Format](#)"

[MS-FSWADF] Microsoft Corporation, "[WebAnalyzer Data File Format](#)"

[MS-FSWCU] Microsoft Corporation, "[WebAnalyzer/Crawler Utility Structure Specification](#)"

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, <http://www.ietf.org/rfc/rfc1321.txt>

[RFC1950] Deutsch, P., and Gailly, J-L., "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996, <http://www.ietf.org/rfc/rfc1950.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

[RFC3548] Josefsson, S., Ed., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003, <http://www.ietf.org/rfc/rfc3548.txt>

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, <http://www.rfc-editor.org/rfc/rfc5234.txt>

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, <http://www.ietf.org/rfc/rfc0793.txt>

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

1.3 Protocol Overview (Synopsis)

This protocol analyzes **search clickthrough** logs and **hyperlink** structures, both analyses contributing to better **ranking** of search results. The protocol is used between one master server and several worker servers in a parallel computing cluster. It is not a traditional client-server protocol, because all protocol servers play the role of protocol client or protocol server in different contexts. Within the master server-worker server relationship however, the master server is the protocol server and the worker server is the protocol client.

The overall task of performing this analysis is managed by the protocol server as a graph of sub tasks. The protocol server is responsible for initiating those discrete sub tasks by protocol clients, keeping track of the locations of data items and status of each subtask.

Before initiating a sub task on a protocol client, the protocol server asserts that all data required to perform the sub task exists on the protocol client. If a file is missing, the protocol server instructs the protocol client that has the file to copy it to the protocol client that requires the file. The protocol server can also instruct protocol clients to delete files that are no longer required.

Protocol clients are unaware of other protocol clients, but when instructed to by the protocol server intercommunicate to exchange data.

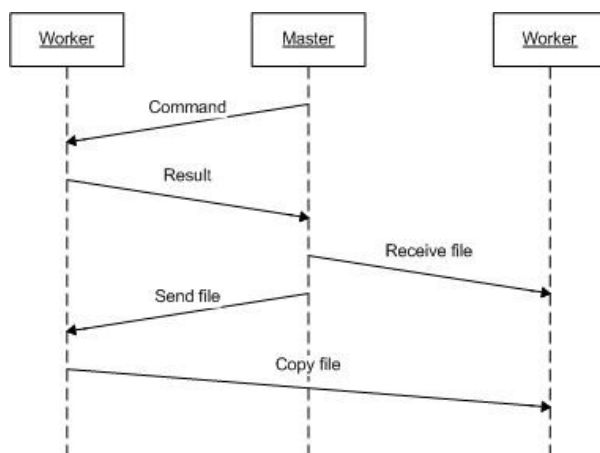


Figure 1: One protocol server and two protocol clients

The preceding figure displays a protocol scenario with one protocol server and two protocol clients. The protocol server sends one command to each of the protocol clients, and collects the results from each. Later, a file is moved between the protocol clients, following instructions sent by the protocol server to the protocol clients to send and receive the file, respectively.

This protocol is symmetric as it does not follow a traditional client-server mode. Instead, both the protocol server and a protocol client send commands to one another.

Messages in this protocol are transmitted as lines of ASCII text.

1.4 Relationship to Other Protocols

This protocol uses TCP/IP for transport, as described in [\[RFC793\]](#). The following diagram shows this protocol in relationship to other protocols:

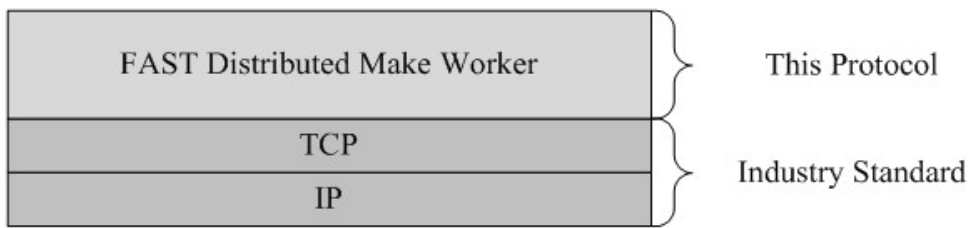


Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is suitable for calculating the connectivity of single **items** in a larger network of interlinked Web items. These calculations are performed in parallel on a distributed computing cluster.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Transport for this protocol MUST be TCP, as specified in [\[RFC793\]](#). Messages MUST be processed in the order they are received, but it is not a requirement that messages are dispatched synchronously. Transport between a protocol server and a protocol client is referred to as a communication channel. Transport between two protocol clients, a **netsink** and a **netsource**, is referred to as a **transfer channel**.

2.2 Message Syntax

Messages in this protocol are case-sensitive commands, represented as sequences of ASCII characters. The first character in each message MUST be the hash character "#", hexadecimal 0x23, and the last character MUST be the ASCII line feed character, hexadecimal 0x0A.

The command name MUST occur after the hash character, followed by a number of tokens, which are the command parameters. Tokens are separated with the ASCII space character " ", hexadecimal 0x20. Consecutive and trailing delimiters are ignored.

Message syntax in protocol commands is specified in the following **Augmented Backus-Naur Form (ABNF)** grammar, as specified in [\[RFC5234\]](#).

```
command-line = "#" 1*command *(FSP (named-arg / base64-encoded-arg))
               *(FSP (arg / base64-encoded-arg)) *FSP LF [*data]

FSP = 1*SP
command = ALPHA / "-" /
named-arg = "#" 1*token "=" 1*token
arg = 1*token
base64-encoded-arg = "@base64:" 1*base64-alphabet *2base64-pad
base64-alphabet = ALPHA / DIGIT / "+" / "/"
base64-pad = "="
token = ALPHA / DIGIT / DQUOTE /
       "{" / "}" / "(" / ")" / "<" / ">" / "/" / "_" /
       "$" / "&" / "%" / "+" / "-" / "=" / "~" / "|" /
       ":" / ";" / "." / "," / "@" / "#" / "\"
data = OCTET
```

Commands, that is, the **command** rule are specified in sections [2.2.4](#) through [2.2.27](#). Named parameters, that is, the **named-arg** rule MUST precede regular parameters, that is, the **arg** rule.

Any named parameter or parameter MUST be **base64** encoded if it contains characters not covered by the **token** rule. Parameters that are covered by the **token** rule can also be encoded. The **base64-alphabet** and **base64-pad** rules are specified in [\[RFC3548\]](#), section 3. There are no line feeds or new lines in the encoded fields, as specified in [\[RFC3548\]](#) section 2.1. Multiple parameters can be contained within one base64 encoded string.

In the following signature, each command is specified with parameters enclosed in angle brackets. Sequences that can be repeated are enclosed in parentheses. Optional sequences are enclosed in square brackets. The command name is the string "command", parameter1 is required, parameter2 can occur one or more times, and parameter3 is optional.

```
#command <parameter1> (parameter2) [parameter3]
```

Because all commands end with the ASCII line feed character, it is omitted for readability in the following sections, except for the three commands in the following table.

Some commands are associated with a data stream which MUST occur immediately after the command. The length of the stream MUST be specified as a command parameter, and the stream MUST be terminated by an ASCII line feed character. The following table specifies the commands that are associated with a data stream.

Command name	Definition
simple_response	section 2.2.23
stderr	section 2.2.25
stdout	section 2.2.26

2.2.1 Simple Types

2.2.1.1 Task Identifiers

Task identifiers MUST be arbitrary size positive integers, represented in base 10 as ASCII encoded strings. They uniquely specify a task running on a protocol client. A task can be a child process, or some other limited processing task that the protocol client has begun processing, but the exact nature of a task is implementation specific.

2.2.1.2 Path Names

Path names MUST be ASCII encoded strings. Path names refer to files or directories on a file system. The path segment separator MUST be the forward slash character "/" (hex 0x2f), or the backslash character "\" (hex 0x5c). Using a mix of the two segment separators in the same path MUST be permitted. Path names can contain **environment variable** references, as specified following the ABNF path name rules.

Paths can be absolute or relative, and relative paths MUST be resolved against the current working directory of the computer that receives a message that contains a path name unless explicitly stated otherwise.

The ABNF rules for path names are specified in the following grammar.

```
path = [drive] [separator] *(segment separator) [segment]
drive = ALPHA ":"
separator = "/" / "\"
segment = string-with-tokens / plain-token

plain-token = 1*(ALPHA / DIGIT / SP / "!" / "#" / "$" / "%" / "&" /
    "'" / "(" / ")" / "+" / "," / "-" / "." /
    ";" / "=" / "@" / "[" / "]" / "^" / "_" /
    "`" / "{" / "}" / "~")
```

Environment variable references are tokens that are embedded in path names. A token contains a variable name that MUST refer to the named variable in the environment of a process.

These references MUST be expanded by replacing the tokens with the contents of the named variables. If the named variable does not exist, the token MUST be left in place unmodified. Recursive references MUST NOT be used.

Expansion can be disabled for portions of the string.

Tokens MUST be specified with a leading ASCII dollar character "\$" (hex 0x24) as an escape character. To use the "\$" character so that it is not expanded as a token, the literal dollar character MUST be encoded by repeating the escape character, as in the string "\$\$". Tokens are further encoded as either suffix or infix tokens, as specified following the token detection ABNF grammar.

The ABNF rules for detecting tokens and quoted portions are specified as follows.

```
string-with-tokens = *restricted *(token / 1*restricted / quoted-text-ended)
                    *(quoted-text-unended / infix-token-unended)
unrestricted = ALPHA / DIGIT / "-" / "_"
restricted = unrestricted / SP / "!" / "#" / "$" / "%" / "&" / "'" /
              "(" / ")" / "+" / "," / "." / ";" /
              "=" / "@" / "[" / "]" / "^" / "`" /
              "~"

QUOTE = "'"
quoted-text-ended = QUOTE *restricted QUOTE ; See QUOTE
quoted-text-unended = QUOTE *restricted ; See QUOTE
token = "$" suffix-token / infix-token /
        escaped-delimiter
suffix-token = 1*unrestricted ; See the following
infix-token-unended = "{" 1*restricted
infix-token = infix-token-unended "}" ; See the following
escaped-delimiter = "$"
```

QUOTE: The single quote character "'" (hex 0x27) MUST disable all expansion for the remainder or portions of the string. If the single quote character is later followed by another single quote character, the single quote characters and the string enclosed within MUST be left unmodified. If a single quote character is not followed by another single quote character, the single quote character MUST be removed, and the text following MUST be left unmodified.

Suffix-token: When the escape character is encountered, the token MUST follow immediately after the escape character, and spans until the next non-alphanumeric character is encountered.

Infix-token: When the escape character is encountered, a token beginning character "{" (hex 0x7b) MUST follow immediately, and spans until a token end character "}" (hex 0x7d) is encountered. If no token end character is encountered, the token MUST be left unmodified, including the escape character and the token beginning character.

2.2.1.3 Transfer Key

The transfer key MUST be composed of a path part and a random part that specify a unique pair of netsink and netsource commands, as specified in section [2.2.15](#) and section [2.2.16](#), respectively. The syntax is specified in the following ABNF grammar.

```
transfer-key = "-" path-part "-" random-part
path-part = path
random-part = 1*DIGIT
```

The **path-part** specifies the local **path**, as specified in section [2.2.1.2](#), to a file to copy to the protocol client that was nominated to be the netsource by receiving the **netsource** message. The random part MUST be a 64 bit pseudo-random number in a base 10 string representation generated by a **pseudo-random number generator (PRNG)**. The PRNG seed state is undefined.

2.2.2 Enumerations

2.2.2.1 Task Type Enumeration

The task type enumeration specifies the type of task for specific message. The enumeration MUST be a string, as specified in the following table.

Value
netsink
netsource
remove-files
shell

2.2.2.2 Task Status Enumeration

The task status enumeration specifies the process state of tasks for specific messages. It MUST be a string, as specified in the following table.

Value
aborted
completed
error

2.2.2.3 Merge Function Enumerations

The merge function enumeration specifies the merge function names, which are specified in the following table.

Merge function names
spmakefileutils.merge_with_global_count
spmakefileutils.merge_with_query
spmakefileutils.merge_with_url
spmakefileutils.normalize_url
wamakefileutils.compute_linkscore_divide_by_linkcount
wamakefileutils.compact_links_and_merge_fromsite
wamakefileutils.cut_fromuris_and_reverse_urihash
wamakefileutils.extend_link_with_freqs
wamakefileutils.filter_intra_and_merge_tosite
wamakefileutils.logcompact_eqrepr

Merge function names
wamakefileutils.filter_sitemap
wamakefileutils.logcompact_links
wamakefileutils.logcompact_no_links
wamakefileutils.logcompact_urieq
wamakefileutils.make_new_urihashmap
wamakefileutils.merge_in_rank
wamakefileutils.merge_repr
wamakefileutils.merge_siterank
wamakefileutils.normalize_link
wamakefileutils.reduce_anchor_and_add_rank
wamakefileutils.reverse_urihash
wamakefileutils.parse_site
wamakefileutils.sum_linkscore_and_new_static_rank

2.2.2.4 Pythontasks Functions Enumerations

The pythontasks function enumeration specifies the pythontasks function names, which are specified in the following table.

Pythontasks function names
_local_copy_func
_local_touch_func
spmakefileutils.file_finder
wamakefileutils.clean_view_build_dir
wamakefileutils.compare_links_and_nolinks
wamakefileutils.dir_creator
wamakefileutils.file_finder
wamakefileutils.remove_coll_directory
wamakefileutils.remove_coll_files
wamakefileutils.remove_directories
wamakefileutils.remove_files_in_filter
wamakefileutils.remove_files_not_in_filter

2.2.2.5 Reduce Functions Enumerations

The reduce functions enumeration specifies the reduce function names, which are specified in the following table.

Reduce function names
spmakefileutils.compact_uris
spmakefileutils.local_count_queries
spmakefileutils.filter_old_urls
spmakefileutils.full_global_count_queries
spmakefileutils.full_local_count_queries
spmakefileutils.logcompact_contentids
spmakefileutils.make_uris_uniq
spmakefileutils.reduce_localcount
spmakefileutils.reduce_urls
spmakefileutils.tokenize_query
wamakefileutils.anchor_weight
wamakefileutils.compact_eqrepr
wamakefileutils.compact_links
wamakefileutils.compact_urieq
wamakefileutils.cut_linkuris
wamakefileutils.cut_urieq
wamakefileutils.do_siterank
wamakefileutils.eqrepr_by_urihash_prep
wamakefileutils.extract_links
wamakefileutils.initial_rank
wamakefileutils.links_by_from_prep
wamakefileutils.make_unique
wamakefileutils.reduce_class
wamakefileutils.urieq_by_class_prep

2.2.2.6 Shell Command Enumerations

The commands enumeration specifies the set of commands that occur in shell messages, as listed in the following table.

Command Name	Section
cobra	2.2.21.1
create_attribute_files	2.2.21.2.1
jsort2.exe	2.2.21.2.2
make_pu_diff	2.2.21.2.3
pupdateclient	2.2.21.2.4
sp_make_pu_diff	2.2.21.2.5
truncate_anchorinfo	2.2.21.2.6
truncate_clickinfo	2.2.21.2.7

2.2.3 Complex Types

2.2.3.1 Dir_size Tuple Structure

The dir_size tuple structure in the following table specifies the number of files and the total size of files contained in a parent directory. The structure is specified in [\[MS-FSWCU\]](#). The Type column refers to types specified in [\[MS-FSWCU\]](#).

Index	Type	Description
0	Integer	Total size as number of bytes of all files contained in the specified directory and its subdirectories. Directories MUST not be counted.
1	Integer	The total number of files in the specified directory and its subdirectories. Directories MUST not be counted.

2.2.3.2 Heartbeat_info Structure

For each member, the following table specifies a name as a string and the type as specified in [\[MS-FSWCU\]](#). All members that are mandatory MUST be included.

Name	Mandatory	Type	Description
id	Yes	Integer	This member is the heartbeat identifier from the request_heartbeat message specified in section 2.2.19 .
pid	Yes	Integer	The process identifier (PID) of the protocol client's process.
disk_mb	No	Long	The available disk space in megabytes on the current working directory of the protocol client's process.
load	No	Integer	This member is reserved and MUST be set to 0.
tmpsize	No	Tuple	This member MUST be a path_dir_size tuple reflecting the temporary directory specified in the request_heartbeat message in section 2.2.19 . The path_dir_size tuple is specified in section 2.2.3.3 .

2.2.3.3 Path_dir_size Tuple Structure

The path_dir_size tuple structure represents a path name and its associated statistics, represented by a dir_size tuple. The tuple structure is specified in [\[MS-FSWCU\]](#). For each element, the following table specifies a fixed index and the type according to [\[MS-FSWCU\]](#). All elements MUST be included.

Index	Type	Description
0	String	This member MUST be a path to the directory
1	Tuple	This member MUST be a dir_size tuple reflecting the temporary directory. The dir_size tuple is specified in section 2.2.3.1 .

2.2.3.4 Directory_listing Array

This **string** contains the contents of a specified root directory. Directories contained within the root directory MUST NOT be included in a directory_listing array. directory_listing arrays are either deep or shallow. For shallow listings, only the files immediately beneath the root directory MUST be included in the directory_listing array. For deep listings, all files contained within the root directory, including those in sub directories, MUST be included.

An entry in a directory_listing array MUST contain the file size in number of bytes and the file name; they MUST be separated by a space " " (hex 0x20), and they MUST be terminated by the ASCII line feed character (hex 0x0A). The ABNF for an entry in a directory listing is specified as follows.

```
dir-listing = *dir-entry
dir-entry = 1*DIGIT SP string-with-tokens LF
```

The string-with-tokens rule is the same as is specified in section [2.2.1.2](#).

2.2.3.5 File_listing Array

This **string** contains file or directory names that are contained in a specified root directory. The directory listing MUST include only the specified directory, and MUST NOT nest to deeper levels. The file names MUST NOT be included if they do not contain a specified string.

An entry in a file_listing array MUST contain the file name terminated by the ASCII line feed character. The ABNF for an entry is specified as follows.

```
file-listing = *file-entry
file-entry = string LF
```

2.2.3.6 Task_info Structure

For each member, the following table specifies a name as a string and the type as specified in [\[MS-FSWCU\]](#). All members that are mandatory MUST be included.

Name	Type	Description
runtime	Floating	The number of seconds that elapsed after a task began processing.
self	String	Specifies the class of a task. It MUST be one of the task classes specified in the following table.

Name	Type	Description
type	String	Specifies the name of a task. It MUST be set to one of the task names specified in the following table.

The following table specifies the mapping between task class and the task name used in the task_info structure.

Task class / self	Task name / type
__main__.remove_file_task	remove-files
__main__.shell_task	shell
__main__.acceptor_task	Netsink

2.2.3.7 Task_info_copy Structure

For each member, the following table specifies a name as a string and the type specified in [\[MS-FSWCU\]](#). All members that are mandatory MUST be included.

Name	Type	Description
copy_bw	Floating	An estimate of the transfer speed at which a file is being transferred between a netsource and a netsink. If an estimate cannot be specified, this field MUST instead be a None field of the None type specified in [MS-FSWCU] section 2.1.4.
progress	Integer	The number of bytes of a file that was copied between a netsource and a netsink.
runtime	Floating	The number of seconds that elapsed after a task began processing.
type	String	Specifies the name of a task. It MUST be set to "netsink" or "netsource".

2.2.3.8 Sortkeyspec String

A sortkeyspec MUST be a string that specifies one or more numbered columns, and how to sort them.

A sortkeyspec MUST begin with the string "-k" immediately followed by a positive base 10 integer which specifies the first selected column, the beginning column. If the sortkeyspec has no further data, it signifies that all columns following the beginning column inclusively MUST be selected.

If a range of columns is selected, a comma character "," (hex 0x2C) MUST follow immediately, again followed by a base 10 integer that specifies the second column, that is, the end column. All columns between the two specified columns inclusively MUST be selected. This syntax MUST also be used to specify a single column. In this case the beginning and end columns MUST be equal.

If a single column is a number, the character "n" (0x6E) MUST immediately follow the end column. In this case the beginning and end columns MUST be equal.

If the columns selected will be sorted in reversed order, the character "r" (hex 0x72) MUST immediately follow.

The preceding rules are summarized in the following ABNF rules:

; Valid combinations: -k1 -k1,1 -k1,1r -k1,5 -k2,2n -k2,2nr

; Combinations that are not valid:

; '-k0' : the first key field is 1, not 0.

; '-k-1' : the first key field is 1, not -1.

; '-k1,2n' : ranged numeric field not allowed, numeric applies to a single column.

SORT-KEY-SPEC = KEY-FIELD [*(SP KEY-FIELD)]

KEY-FIELD = "-k" START (CLOSED / OPEN)

OPEN = ["r"]

CLOSED = ", " END ["nr" / "n" / "r"] ; if n is present, START MUST be equal to END.

START = NUMBER

END = NUMBER

NUMBER = %x31-39 ; 1-9

2.2.3.9 Tableinputspec String

A tableinputspec MUST be a string in which three values are concatenated with an ASCII colon characters ":" (hex 0x3A). The values MUST be as specified in the following table.

Value name	Definition
filename	MUST be the name of a file containing data.
numcolumns	MUST be an integer in base 10 that specifies the number of data columns in the data file named by the filename field. This value MUST be greater than 0.
key	MUST be an integer in base 10 that specifies the numbered column that contains keys in the data file whose name is contained in the filename field. This value MUST be greater than 0, and less than or equal to the numcolumns field.

2.2.3.10 Tableoutputspec String

A tableoutputspec MUST be a string in which three values are concatenated with an ASCII colon characters ":" (hex 0x3A). The values MUST be as specified in the following table.

Value name	Definition
filename_prefix	MUST be a name prefix shared by files where data will be written.
key	MUST be an integer in base 10 that is greater than or equal to zero and specifies the numbered column that contains keys in the data file named by the filename field.
split_factor	MUST be an integer in base 10 that is greater than or equal to zero, and specifies the number of files into which to split the data.

2.2.3.11 Target_name String

A target_name MUST be a string in which three or four values are concatenated with an ASCII dot character "." (hex 0x2E).

Value name	Definition
prefix	MUST be a string. Optional.
collection	MUST be a string that specifies a content collection name in the system.
timestamp	MUST be an integer in base 10 representation.
name	MUST be a string.

2.2.4 Abort Command

This message specifies the task identifier of the task to terminate processing.

```
#abort <task-ID>
```

Task-ID: MUST specify the identifier of the task to terminate.

2.2.5 CD Command

This message MUST contain a directory parameter to use as a current working directory.

```
#cd <new-directory>
```

New-directory: This parameter MUST be a path name as specified in section [2.2.1.2](#), and MUST specify the directory to make the current working directory.

2.2.6 Close-stdin Command

This message MUST close the standard input stream of a task. This message MUST have one parameter that specifies the task.

```
#close-stdin <task-ID>
```

Task-ID: MUST specify the task identifier of the task which standard input stream is to be closed

2.2.7 Disk_limit Command

This message specifies the minimum free disk space required for proper operation.

```
#disk_limit <number-of-mb>
```

Number-of-mb: The amount of data required specified as an arbitrary size positive integer and represented as an ASCII string. This parameter MUST be expressed as a multiple of megabytes, a megabyte equaling 2^{20} bytes.

2.2.8 Error Message

This message contains information about the error condition when a task has failed in a protocol client. This message MUST contain a task identifier. The remainder of this message is the *info*

parameter, in which implementers add any human-readable data to specify information about the error condition that occurred.

```
#error <task-ID> [info]
```

Task-ID: MUST specify the task identifier of the task that failed.

Info: This optional parameter is undefined, and MUST NOT be parsed. It is included only to specify human-readable context about the error that occurred.

2.2.9 Finish Message

This message specifies status information about a task that finished processing. This message MUST contain at least three parameters: task identifier, task type and task status.

```
#finish <task-ID> <task-type> <task-status> [auxiliary-information] [info]
```

Task-ID: MUST specify the task identifier of the task that began on this communication channel.

Task-type: Specifies the type of the task that finished processing. The fields allowed for this parameter is specified in section [2.2.2.1](#).

Task-status: Specifies the outcome of the task that finished processing. The fields allowed for this parameter are specified in section [2.2.2.2](#).

Auxiliary-information: The content of this message varies depending on the *task-type* and *task-status* parameters. The following table specifies the possible values. Integer values MUST be arbitrary size integers in base 10 represented as a string, whereas task identifiers are as specified in section [2.2.1.1](#). For some task types, the parameter is undefined and MUST be omitted. In those cases, parameter value and parameter type are specified in the following table with the value "N/A".

task type	task status	parameter value	type
netsink	completed	0	integer
	aborted	N/A	N/A
	Error	N/A	N/A
netsource	completed	0	integer
	aborted	N/A	N/A
	Error	N/A	N/A
remove-files	completed	N/A	N/A
	aborted	task identifier	task identifier
shell	completed	return code	integer
	aborted	1	integer

Info: This parameter is undefined, and MUST NOT be parsed by the receiver. It specifies human-readable context about an error that occurred, in context with the *task-status* and *auxiliary-information* parameters.

2.2.10 Finish-tasks Command

This message directs the receiver to finish all processing. This message has no parameters.

```
#finish-tasks
```

2.2.11 Finish-tasks-ack Message

This message means that the protocol client finished all processing. This message MUST contain one parameter.

```
#finish-tasks-ack <finish-tasks-ack>
```

Finish-tasks-ack: This contains the string "finish-tasks-ack".

2.2.12 Forget Command

This message MUST contain a task identifier parameter. Upon receiving this message the protocol client MUST clear all state associated with the specified task identifier.

```
#forget <task-ID>
```

Task-ID: MUST specify the task identifier of a task that began on this communication or transfer channel

2.2.13 Heartbeat Message

This message indicates that the protocol client is alive and running normally. This message has one parameter. The parameter is a collection of named variables encoded using the protocol specified in [\[MS-FSWCU\]](#).

```
#heartbeat <heartbeat-info>
```

Heartbeat-info: A structure of named variables. The structure MUST be encoded as a Dictionary data type, as specified in [\[MS-FSWCU\]](#). The members of the structure are specified in section [2.2.3.2](#).

2.2.14 Info Message

This message sends status information for a human operator. This message MUST have an *info* parameter.

```
#info <info>
```

Info: MUST be a string with a human readable message.

2.2.15 Netsink Command

This message specifies that the protocol client is the netsink in the netsink-netsource transaction. The netsink receives a file from the netsource. The message MUST contain four parameters.

```
#netsink <task-ID> <destination-path> <transfer-key> <compress>
```

Task-ID: MUST specify the task identifier for a set of file transfer messages.

Destination-path: An absolute path that MUST exist on the netsink, as specified in section [2.2.1.2](#).

Transfer-key: A unique transfer key as specified in section [2.2.1.3](#).

Compress: This parameter controls compression during file transfer. It MUST contain the string "compress" or the string "nocompress".

2.2.16 Netsource Command

This message specifies that the protocol client is the netsource in the netsink-netsource transaction. The netsource sends a file to the netsink. This message MUST contain six parameters.

```
#netsource <task-ID> <source-path> <receiver-hostname> <receiver-port> <transfer-key>  
<compress>
```

Task-ID: MUST specify the task identifier for a set of file transfer messages

Source-path: An absolute path that MUST exist on the protocol client. Embedded environment variable references as specified in section [2.2.1.2](#) are permitted.

Receiver-hostname: A **Domain Name System (DNS)** host name of the receiver protocol client that MUST be resolvable in DNS.

Receiver-port: MUST be a TCP port number on the receiver protocol client.

Transfer-key: A unique transfer key as specified in section [2.2.1.3](#).

Compress: This parameter controls compression during file transfer. It MUST contain either the string "compress" or the string "nocompress".

2.2.17 Poll-task Command

This message requests status information for a task. This message MUST contain a *task-ID* parameter.

```
#poll-task <task-ID>
```

Task-ID: MUST specify the task identifier of the task that has begun processing.

2.2.18 Remove-files Command

This message MUST contain a *task-ID* parameter, and one or more files to remove.

```
#remove-files <task-ID> [inputdir=<directory>] <files>
```

Task-ID: MUST contain the task identifier, as specified in section [2.2.1.1](#).

Directory: If present, MUST be a path to a directory. The path MUST be formatted as specified in section [2.2.1.2](#). This directory MUST NOT be removed and files in this directory MUST remain unchanged as specified in section [3.3.5.1.1](#).

Files: One or more file paths to be removed. Multiple paths MUST be separated by the ASCII space character " " (hex 0x20). Paths MUST be relative to the current working directory of the receiver protocol client. The paths MUST be formatted as specified in section [2.2.1.2](#).

2.2.19 Request_heartbeat Command

This message requests a **heartbeat** message from the protocol client or netsink. This message MUST include the *heartbeat-ID* parameter. The protocol server or netsource MUST include the temporary directory parameter asking for the size of the temporary directory. If the protocol server or netsource needs to uniquely specify the response from the protocol client or netsink, a *heartbeat-ID* parameter MUST be included.

```
#request_heartbeat [#tmp=<temporary-directory>] <heartbeat-ID>
```

Temporary-directory (optional): A path to a temporary directory on the protocol client or netsink that the protocol client or netsink MUST use to respond to the heartbeat message. The path MUST be a string, and MUST be a relative path. The path MUST be formatted as specified in section [2.2.1.2](#).

Heartbeat-ID: MUST be an arbitrary size positive integer represented as a base 10 ASCII encoded string that the receiver protocol client MUST use when responding with the **heartbeat** message.

2.2.20 Setenv Command

This message specifies a name-value pair to be interpreted as an environment variable. The message MUST contain the name and the value parameters. The name-value pair MUST be as specified in the following table.

```
#setenv <name> <value>
```

Name: MUST be a string that specifies the name of the variable to set

Value: MUST be a string that specifies the value of the variable to set.

Environment variable name	Value
LANG	C
LC_ALL	C

2.2.21 Shell Command

This message initiates a task on a protocol client on behalf of a protocol server, and has a varying number of parameters.

```
#shell <umask> <task-ID> cmd /C <command> [command-args]
```

Umask: This parameter MUST be set to the string "#umask=54". This parameter MUST be ignored.

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Cmd /C: This parameter MUST be ignored.

Command: This parameter MUST be a command specified in section [2.2.2.6](#).

Command-args: Any number of subsequent tokens in the command line. They are dependent on the command parameter, and are specified in the following subsections.

2.2.21.1 Cobra Command

The following commands MUST all contain the string "\$FASTSEARCH\lib\python2.5\" as a prefix.

The following table specifies the **cobra** commands. The commands all begin with the stated prefix, however the prefix is omitted in the table and section titles for readability.

Cobra command name	Section
merge.pyc	2.2.21.1.1
pythontasks.pyc	2.2.21.1.2
reduce.pyc	2.2.21.1.3
split.pyc	2.2.21.1.4

2.2.21.1.1 Merge.pyc Command

A message containing a merge.pyc command MUST be as the following:

```
#shell umask=54 <task-ID> cmd /C cobra $FASTSEARCH\lib\python2.5\merge.pyc (-i <inputspec>) -  
o <outputspec> -l <merge-function> [reduce]
```

Inputspec: There MUST be 2 or 3 inputspec parameters. Each inputspec parameter MUST begin with the string "-" immediately followed by an ASCII space character " " (hex 0x20), which is again immediately followed by a tableinputspec value, as specified in section [2.2.3.9](#).

Outputspec: MUST be a tableoutputspec, as specified in section [2.2.3.10](#).

Merge-function: MUST be one of the function names specified in section [2.2.2.3](#).

Reduce: If present, this parameter MUST be set to the string "-r".

2.2.21.1.2 Pythontasks.pyc Command

A message containing a pythontasks.pyc command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C cobra $FASTSEARCH\lib\python2.5\pythontasks.pyc  
<pythontasks-function> <parameters>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Pythontasks-function: MUST be one of the function names specified in section [2.2.2.4](#).

Parameters: The parameters for the **pythontask** functions are specified in section [2.2.21.1.2.1](#).

2.2.21.1.2.1 pythontasks functions and parameters

The following tables specify the parameters for the **pythontask** functions. All parameters named with a trailing `_path` MUST be a path name as specified in section [2.2.1.2](#). All other parameters are strings, unless otherwise stated.

The functions specified in the following table MUST accept at least one parameter. The *directories* parameters MUST be directory paths.

Pythontasks function name	1st N parameters
wamakefileutils.remove_directories	directories
wamakefileutils.dir_creator	directories

The function specified in the following table MUST accept exactly two parameters. The *permissions* parameter MUST either be the string "None", or be a base 10 represented string.

Pythontasks function name	1st parameter	2nd parameter
_local_touch_func	file-path	permissions

The functions specified in the following table MUST accept exactly two parameters. The *dir-path* parameter MUST be a path to a directory, and the *collectionname* parameter MUST be an alphanumeric string.

Pythontasks function name	1st parameter	2nd parameter
wamakefileutils.remove_coll_directory	dir-path	collectionname
wamakefileutils.remove_coll_files	dir-path	collectionname

The functions that are specified in the following table MUST accept exactly one initial parameter, and at least one final parameter. The *dir-path* parameter MUST be a path, and the *prefixes* parameter MUST be a sequence of path prefixes.

Pythontasks function name	1st parameter	Last N parameter
wamakefileutils.remove_files_in_filter	dir_path	prefixes
wamakefileutils.remove_files_not_in_filter	dir_path	prefixes

The function that is specified in the following table MUST accept exactly three parameters. The *permissions* parameter MUST either be the string "None", or be a string represented base 10 integer.

Pythontasks function name	1st parameter	2nd parameter	3rd parameter
_local_copy_func	Source-path	Destination-path	permissions

The function that is specified in the following table MUST accept exactly three parameters. The *prefix* parameter MUST be a path prefix, and the *index* and *split-factor* parameters MUST be string represented base 10 integers.

Pythontasks function name	1st parameter	2nd parameter	3rd parameter
wamakefileutils.compare_links_and_nolinks	prefix	index	split-factor

The function that is specified in the following table MUST accept two initial parameters, and zero or more final parameters.

Pythontasks function name	1st parameter	2nd parameter	Last N parameters
wamakefileutils.clean_view_build_dir	build-path	input-path	prefixes

The functions that are specified in the following table MUST accept three initial parameters, and one or more final parameters. The *linkdump-path* parameter MUST be a path name, the *collectionname* MUST be a string of alphanumeric characters, the *output-prefix* parameter MUST be a path prefix, and the *filenames* parameter MUST be a sequence of file paths.

Pythontasks function name	1st parameter	2nd parameter	3rd parameter	Last N parameters
spmakefileutils.file_finder	linkdump-path	collectionname	output-prefix	filenames
wamakefileutils.file_finder	linkdump-path	collectionname	output-prefix	filenames

2.2.21.1.3 Reduce.pyc Command

A message containing a reduce.pyc command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C cobra $FASTSEARCH\lib\python2.5\reduce.pyc -i <inputspec> -
o <outputspec> -l <reduce-function> [--funcvars=<funcvars>]
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Inputspect: MUST be a tableinputspect value as specified in section [2.2.3.9](#).

Outputspect: MUST be a tableoutputspect value as specified in section [2.2.3.10](#).

Reduce-function: MUST be one of the function names specified in section [2.2.2.5](#), if present.

Funcvars: If present, MUST be a Dictionary as specified in [\[MS-FSWCU\]](#) that is compressed using the zlib algorithm specified in [\[RFC1950\]](#), and again encoded in base64 using the alphabet specified in [\[RFC3548\]](#) section 4.

2.2.21.1.4 Split.pyc Command

A message containing a split.pyc command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C cobra $FASTSEARCH\lib\python2.5\split.pyc <inputfile> (-o
<outputspect>) [--splitfunction2 <splitfunction>]
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Inputfile: MUST be a path to a file.

Outputspec: There MUST be 1 or more outputspec parameters. Each outputspec parameter MUST be a tableoutputspec value, as specified in section [2.2.3.10](#), and preceded by the string "-o".

Splitfunction: If present, MUST be the string "wamakefileutils.long_split" or the string "spmakefileutils.long_split".

2.2.21.2 Other Shell Commands

The following are **shell** commands that are not **shell cobra** commands. They have been separated from the shell cobra commands for readability..

Command Name	Section
create_attribute_files	2.2.21.2.1
jsort2.exe	2.2.21.2.2
make_pu_diff	2.2.21.2.3
pupdateclient	2.2.21.2.4
sp_make_pu_diff	2.2.21.2.5
truncate_anchorinfo	2.2.21.2.6
truncate_clickinfo	2.2.21.2.7

2.2.21.2.1 Create_attribute_files Command

A message containing a create_attribute_files command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C create_attribute_files -i <inputfile> -o <outputfile>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Intputfile: MUST be the path to an already existing file.

Outputfile: MUST be the path prefix.

2.2.21.2.2 Jsort2.exe Command

A message containing a jsort2.exe command MUST be specified as follows.

```
#shell #umask=54 <task-ID> cmd /C jsort2.exe -S <size> -T <tmpdir> [merge] [unique]
[sortkeyspec] <inputfiles> -o <output-file>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Size: This MUST set an upper limit for memory usage for the sorting operation, as specified in section [3.7.5.2](#). The value MUST be specified in megabytes as an ASCII encoded base 10 string and MUST be terminated by the character "M". The default value in the system is "200M".

Tmpdir: This parameter MUST be set to the dot character "." (hex 0x2E). This parameter MUST be ignored.

Merge: MUST be the string "-m", if present.

Unique: MUST be the string "-u", if present.

Sortkeyspec: This parameter specifies what columns should be selected, and MUST be as specified in section [2.2.3.8](#). There can be zero or more *sortkeyspec* parameters.

Inputfile: One or more input files, separated by spaces. At least one MUST be present.

Output-file: This parameter MUST specify an output file. This parameter MUST be present.

2.2.21.2.3 Make_pu_diff Command

A message containing a *make_pu_diff* command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C make_pu_diff -a <oldfile> -b <newfile> -m <mainfile> -o  
<outputfile> -c <collectionname> -n <isnew>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Oldfile: MUST be a path to a file.

Newfile: MUST be a path to a file.

Mainfile: MUST be a path to a file.

Outputfile: MUST be a path to a file.

Collectionname: MUST be string that specifies a content collection name in the system.

isnew: MUST be either the string 0 or 1.

2.2.21.2.4 Pupdateclient Command

A message containing a *pupdateclient* command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C pupdateclient <inputfile> <batchsize> <collectionname>  
<timeout> <id>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Inputfile: MUST be a path to a file.

Batchsize: MUST be an integer represented in base 10. This parameter MUST be ignored.

Collectionname: MUST be a string that specifies a content collection name in the system.

Timeout: MUST be an integer represented in base 10. This parameter MUST be ignored.

Id: MUST be a string. This parameter MUST be ignored.

2.2.21.2.5 Sp_make_pu_diff Command

A message containing a *sp_make_pu_diff* command MUST be as the following:


```
#shell #umask=54 <task-ID> cmd /C sp_make_pu_diff -a <oldfile> -b <newfile> -m <mainfile> -o  
<outputfile> -c <collectionname> [-n <isnew>]
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Oldfile: MUST be a path to a file.

Newfile: MUST be a path to a file.

Mainfile: MUST be a path to a file.

Outputfile: MUST be a path to a file.

Collectionname: MUST be string that specifies a content collection name in the system.

Isnew: This optional parameter MUST be either the string 0 or 1.

2.2.21.2.6 Truncate_anchorinfo Command

A message containing a truncate_anchorinfo command MUST be as the following:

```
#shell #umask=54 <task-ID> cmd /C truncate_anchorinfo -a <oldfile> -b <newfile> -m <mainfile>  
-o <outputfile> -c <collectionname>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Oldfile: MUST be a path to a file.

Newfile: MUST be a path to a file.

Mainfile: MUST be a path to a file.

Outputfile: MUST be a path to a file.

Collectionname: MUST be a string that specifies a content collection name in the system.

2.2.21.2.7 Truncate_clickinfo Command

A message that contains a truncate_clickinfo command MUST be specified as follows.

```
#shell #umask=54 <task-ID> cmd /C truncate_clickinfo -a <oldfile> -b <newfile> -m <mainfile>  
-o <outputfile> -c <collectionnames>
```

Task-ID: MUST contain a task identifier, as specified in section [2.2.1.1](#).

Oldfile: MUST be a path to a file.

Newfile: MUST be a path to a file.

Mainfile: MUST be a path to a file.

Outputfile: MUST be a path to a file.

Collectionnames: MUST be a string of comma-separated values. Each value MUST specify a content collection name in the system.

2.2.22 Simple_command Command

This message initiates a task in a protocol client on behalf of a protocol server. The message includes a command line with a command and one or more parameters. The commands are specified in the following table.

Command name	Purpose
create_builddirs	Create the specified directories.
create_builddirs_no_tmp	Create the specified directories.
find_builddirs	List the files in the specified directory.
link_to_file	Create a hard link to the specified file.
markfile	Mark the specified file as read-only.
remove	Remove the specified file.
touch	Create an empty file in the specified location.

The command has two mandatory parameters that **MUST** be set. In addition, one or more parameters **MUST** follow, depending on the contents of the command parameter.

```
#simple_command <task-ID> <command> [command-args]
```

Task-ID: **MUST** contain a task identifier, as specified in section [2.2.1.1](#).

Command: For each **simple_command** message a specified set of parameters **MUST** be used. The commands are specified in the following table.

Command-args: Any number of parameters to the command parameter. The parameters **MUST** all be path names, as specified in section [2.2.1.2](#). Multiple parameters **MUST** be separated by a single space character (hex 0x20). The number of parameters and their names are specified in the following table.

Command name	Parameter 1	Parameter 2	Parameter 3	Parameter 4
create_builddirs	<i>Parent-dir</i>	<i>build-dir</i>	<i>input-dir</i>	<i>tmp-dir</i>
create_builddirs_no_tmp	<i>parent-dir</i>	<i>build-dir</i>	<i>input-dir</i>	N/A
find_builddirs	<i>parent-dir</i>	<i>path-token</i>	N/A	N/A
link_to_file	<i>source-file</i>	<i>dest-file</i>	N/A	N/A
markfile	<i>file</i>	N/A	N/A	N/A
remove	<i>file</i>	<i>file</i>		
touch	<i>file</i>	N/A	N/A	N/A

The remove command **MUST** accept one or more parameters. Its parameter **MUST** be a file, not a directory.

2.2.23 Simple_response Message

This message reports the completion of a **simple_command**, and MUST contain the *task-ID* and *number-of-bytes* parameters that are specified as follows.

A data stream, whose length is specified by the *number-of-bytes* parameter, MUST follow immediately after this message.

```
#simple_response <task-ID> <number-of-bytes>%0A<data>%0A
```

Task-ID: MUST specify the identifier of a task that a previous **simple_command** message requested to process.

Number-of-bytes: An integer represented in base 10 as an ASCII encoded string which specifies the length of a data stream MUST immediately follow this message. The data stream MUST contain a trailing ASCII line feed (hex 0x0A) character, so this number MUST be one less than the actual length of the stream.

Data: The use of this parameter is implementation dependent. The format of the data stream is specified in the following table.

Command name	Data
create_builddirs	The string result for this message MUST be two <i>directory_listing</i> arrays, specified in section 2.2.3.4 , concatenated together. The first MUST be a shallow listing of the <i>input_dir</i> parameter. The second MUST be a deep listing of the <i>build_dir</i> parameter.
create_builddirs_no_tmp	The same as for <i>create_builddirs</i> .
find_builddirs	The string result for this message MUST be a <i>file_listing</i> array, as specified in section 2.2.3.5 .
link_to_file	An empty string (""). The <i>number-of-bytes</i> parameter MUST be 0.
markfile	The result string MUST be set to the number of bytes the file contains, represented as a base 10 integer in ASCII encoding.
remove	An empty string (""). The <i>number-of-bytes</i> parameter MUST be 0.
touch	An empty string (""). The <i>number-of-bytes</i> parameter MUST be 0.

2.2.24 Slot Message

This message is the response to a message sent previously by the protocol server. The message MUST contain the task identifier in the *task-ID* parameter as was received by the protocol client in the originating message. Further, it MUST contain the task type in the *task-type* parameter, and a *task-state* parameter if that is required for the task type.

```
#slot <task-type> <task-ID> [task-state]
```

Task-type: This parameter MUST contain the command type for the preceding message that was received. The value of this parameter based on this relation is specified in section [2.2.2.1](#).

Task-ID: This MUST be the task identifier that was contained in the message sent previously by the protocol server.

Task-state: This MUST be the task state, if any, depending on the command type. It MUST be set as specified in the last column in the following table. If the value is undefined, it MUST be omitted.

Command	Message definition	Task-state
netsource	section 2.2.16	The size of the file specified by the <i>source-path</i> parameter of the netsource message. The size MUST be specified in bytes as a base 10 decimal integer in a ASCII encoded string.
netsink	section 2.2.15	The TCP port number set up for the netsink, see section 3.4.5 .
shell	section 2.2.21	The process identifier (PID) of process that has begun processing, see section 3.7 .
remove-files	section 2.2.18	Not specified.

2.2.25 Stderr Message

This message contains the standard error stream from a process. This message MUST contain a *task-ID* and a *number-of-bytes* parameter.

A data stream, whose length is specified by the *number-of-bytes* parameter, MUST follow immediately after this message.

```
#stderr <task-ID> <number-of-bytes>%0A<data>%0A
```

Task-ID: MUST specify the identifier of the task associated with the subsequent data stream.

Number-of-bytes: A 32-bit signed integer represented in base 10 which specifies the length of a data stream that will immediately follow this message. This stream MUST contain a trailing ASCII line feed character, so this number MUST be one less than the actual length of the stream.

Data: This parameter MUST be ignored. The parameter is a stream of octets, or bytes, which values are between 0x00 and 0xFF.

2.2.26 Stdout Message

This message contains the standard output stream from a process. This message MUST contain a *task-ID* and a *number-of-bytes* parameter.

A data stream, whose length is specified by the *number-of-bytes* parameter, MUST follow immediately after this message.

```
#stdout <task-ID> <number-of-bytes>%0A<data>%0A
```

Task-ID: MUST specify the identifier of the task associated with the subsequent data stream.

Number-of-bytes: A non-negative arbitrary size integer represented in base 10 which specifies the length of a data stream that will immediately follow this message. This stream MUST contain a

trailing ASCII line feed character, so this number MUST be one less than the actual length of the stream.

Data: This parameter MUST be ignored. The parameter is a stream of octets, or bytes, whose values are between 0x00 and 0xFF.

2.2.27 Task-info Message

This message contains information about a task. This message MUST contain a *task-ID* and a *task-info* parameter, as specified in the following rule.

```
#task-info <task-ID> <task-info>
```

Task-ID: Must specify the identifier of the task for which the *task-info* was collected.

Task-info: This parameter MUST contain a structure of named variables encoded as a Dictionary data type, as specified in [\[MS-FSWCU\]](#). The structure MUST be either the `task_info` structure specified in section [2.2.3.6](#), or the `task_info_copy` structure specified in section [2.2.3.7](#).

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is specified to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Tables: On protocol clients, data is stored in tables. Tables contain records as rows and attributes as columns. Each table has a schema that specifies the set of attributes for each record in terms of type and name, as specified in [\[MS-FSWADF\]](#) and [\[MS-FSSPRDF\]](#). Only the string and integer types are specified.

Tables MUST be stored as ASCII streams in files, where each record MUST be represented by a sequence of field values separated by the ASCII space character (hex 0x20). Each record MUST be terminated by the ASCII line feed character (hex 0x0A). Tables can span across multiple physical files. There are five functions that are performed on tables, specified in section [3.3.5.1.2.1](#). Each function accepts one or more tables as input, which are referred to as *input tables*. Each function produces one or more tables, which are referred to as *output tables*.

Build: The super task referred to as a build is composed of multiple discrete tasks. The tasks depend on data, which can be the output of other tasks. The protocol server MUST have full knowledge about the interdependencies in terms of input and output, and the location of the data during task processing.

Task: The protocol server controls task processing on the protocol client, and MUST generate a process request based on the build, the available protocol clients, and the initial location of the data.

The protocol server MUST assign a task identifier to a task when the task is initiated. The task identifier is sent to the protocol client that processes the task. This task identifier MUST be used by both the protocol server and protocol client when referring to a specific task in the exchange of subsequent messages.

Data Distribution: The initial distribution of data to the protocol clients is implementation-specific. However, the protocol server can move data between protocol clients using the *netsource* and *netsink* commands, as specified in sections [2.2.16](#) and [2.2.15](#) respectively.

Communication Channel: The protocol server MUST establish a TCP connection to each protocol client to use as a communication channel. This communication channel is used for issuing commands in both directions. For some messages the communication channel is also used to send data to the protocol server, in which case the data occurs immediately after the message.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Messages MUST be processed in the order they are received. Messages MUST NOT be sent or received synchronously, so the protocol server and protocol clients can send multiple messages while awaiting a receipt of a response.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Master Protocol Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is specified to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Build: An overall task to be performed. It is composed of a set of possibly interdependent smaller tasks to process in some order. The protocol server controls the overall task and splits it into smaller subtasks. The protocol server processes the subtasks in a suitable order by assigning them to protocol clients. Independent subtasks are processed in parallel.

3.2.2 Timers

3.2.2.1 Heartbeat Timeout Timer

This timer measures the time required for a protocol client to respond to a **request_heartbeat** message, as specified in section [2.2.19](#). A timeout MUST occur if the protocol client does not respond within 900 seconds of the sending of the message.

3.2.2.2 Heartbeat Timer

This timer measures the time between sending **heartbeat** messages, as specified in section [2.2.13](#). The default is 15 seconds, and the value MUST be less than 60 seconds.

3.2.2.3 Build Timeout Timer

The build timer measures the time required for the build to finish processing. The build timeout is implementation-specific.

3.2.3 Initialization

The protocol server MUST begin the build process by getting a list of available protocol clients. How this is done is not specified by this protocol. One possibility is to call the **GetModuleList** method with the string parameter "fdmworker", as specified in [\[MS-FSCX\]](#).

The protocol server sets up a TCP/IP connection to each protocol client. The protocol clients by default listen on the port number **base port** + 310.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Starting Build

The protocol server MUST begin the build process by sending a **cd** message to each protocol client. This message tells the protocol client which local directory to use as working directory during the build. All input files required by commands are assumed to be available in this directory and all files produced by commands are written there. Section [2.2.5](#) specifies the message format.

During the build, the protocol server MUST maintain the list of files available in the working directory for each protocol client.

If a file is no longer needed by the specified protocol client, the protocol server MUST send a **remove-files** message to delete the file. Section [2.2.18](#) specifies the message format.

3.2.4.2 Starting Command on Worker Protocol Client

When the protocol server begins processing a command on a protocol client the protocol server MUST make sure any files required as input for the command are available on the protocol client. The files MUST be present in the working directory set by the **cd** message during build startup.

The input files MUST either already exist on the protocol client or they MUST exist on another protocol client. If they already exist, they MUST have been produced by a previous command message or they MUST have been created by an implementation specific method beyond the scope of this document.

If an input file is present on a different protocol client, the protocol server MUST transfer the file by a netsink-netsource transaction as specified in section [3.2.5](#).

If the file is not present on another protocol client, and if the file has not been produced locally, the protocol server MUST assume the file exists.

The protocol server MUST keep track of all output files produced by all commands during a build. In this way the protocol server can transfer files from one protocol client to another as required during a build.

3.2.4.3 Sending Remove-files Command

If the protocol server deletes a file on a protocol client it MUST send a **remove-files** message, as specified in section [2.2.18](#). The protocol client MUST reply with a **slot** message, as specified in section [2.2.24](#). When the file has been deleted the protocol client MUST send a **finish** message, as specified in section [2.2.9](#). The protocol server replies with a **forget** message, as specified in section [2.2.12](#).

3.2.4.4 Sending Shell Command

When the protocol server processes a command on a protocol client it MUST send a shell message. Section [2.2.21](#) specifies the shell message format. The protocol client MUST reply with a **slot** message, as specified in section [2.2.24](#), and begin to process the command. When the command finishes, the protocol client MUST send a **finish** message, as specified in section [2.2.9](#), back to the protocol server and the protocol server MUST reply with a **forget** message, as specified in section [2.2.12](#).

3.2.4.5 Sending Abort Command

If the protocol server terminates a command that was processing on the protocol client, it MUST send an **abort** message, as specified in section [2.2.4](#). The protocol client MUST reply with a **finish** message, as specified in section [2.2.9](#).

3.2.4.6 Sending Finish-tasks Message

When the build finishes processing, the protocol server server sends a **finish-tasks** message which format is specified in section [2.2.10](#). The protocol server MUST wait for all protocol clients to send a **finish-tasks-ack** message, as specified in section [2.2.11](#).

3.2.4.7 Sending Setenv Command

The protocol server MUST send a **setenv** command to a protocol client to process subsequent tasks in the build process in the specified environment on the protocol client. The message format is specified in section [2.2.20](#).

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Info Message

This message is used to convey informational data for logging purposes, and can contain any number of parameters. It can contain an informational version string, but it MUST NOT be used for versioning or capability negotiation. The message format is specified in section [2.2.14](#).

3.2.5.2 Finish Message

This message, as specified in section [2.2.9](#), specifies that a protocol client has finished the task associated with the *task-ID* parameter. The protocol server MUST reply with a **forget** message, as specified in section [2.2.12](#). The *task-ID* parameter for the **forget** message MUST be the *task-ID* parameter from the **finish** message.

3.2.5.3 Finish-tasks-ack Message

This message, as specified in section [2.2.11](#), is a reply to a previously sent **finish-tasks** message, as specified in section [2.2.10](#). The protocol server can conclude that the **finish-tasks** message completed.

3.2.5.4 Heartbeat Message

The protocol client sends this message as a response to a **request_heartbeat** message to specify that it is responding, as specified in section [2.2.19](#). The protocol server MUST reset the associated protocol client heartbeat timeout, as specified in section [3.2.2.1](#).

3.2.5.5 Netsink-Netsource Transaction

The protocol server can transfer a file from one protocol worker to another using the **netsink** and **netsource** messages, as specified in sections [2.2.15](#) and [2.2.16](#) respectively, in a netsink-netsource transaction. The protocol server MUST start by sending a **netsink** message to a protocol client, a netsink. The file will be transferred to the netsink. The netsink replies with a **slot** message, as specified in section [2.2.24](#), upon success or an **error** message upon failure, as specified in section [2.2.8](#). In the case of failure, the protocol server MUST abort the current file transfer. Upon receiving a **slot** message, the protocol server sends a **netsource** message, as specified in section [2.2.16](#), to another protocol client, a netsource that has the file.

If the file transfer succeeds both the netsink and the netsource replies with a **finish** message with *task-status* "completed", as specified in section [2.2.9](#). At this point the protocol server can conclude that the file has been transferred to the netsink. If the transfer fails, the netsink or netsource replies with a **finish** message with *task-status* "error". If the netsink fails the protocol server MUST abort the file transfer by sending an **abort** message, as specified in section [2.2.4](#), to the netsource, and vice versa if the netsource fails.

3.2.5.6 Simple_response Message

This message is a response to a previous **simple_command** message, as specified in section [2.2.22](#).

The protocol server MUST inspect the *number-of-bytes* parameter and read this many bytes from the communication or transfer channel associated with the protocol client, as specified in section [2.2.23](#). The extra payload contains any output from the command in the **simple_command** message.

3.2.5.7 Stderr Message

The **shell** command, as specified in section [2.2.21](#), has failed for at least one protocol client, so the protocol server MUST receive the data stream that occurs immediately after the message. The length of the data stream is specified in the *number-of-bytes* parameter, and the task that receives the data stream is specified in the *task-ID* parameter.

The stream is terminated by a trailing ASCII line feed character (hex 0x0A), so the number of bytes to read from the stream is one greater than the *number-of-bytes* parameter.

The content of the stream is the standard error stream of a child process that began processing in a shell task on the protocol client.

Upon receiving a **stderr** message, as specified in section [2.2.25](#), the protocol server MUST assume that the shell task failed.

3.2.5.8 Stdout Message

The protocol server MUST receive a data stream immediately following the message. The length of the data stream is specified in the *number-of-bytes* parameter, and the task that receives the data stream is specified in the *task-ID* parameter.

The stream is terminated by a trailing ASCII line feed character (hex 0x0A), so the number of bytes to read from the stream is one greater than the *number-of-bytes* parameter.

The content of the stream is the standard output stream of a child process that began processing in a shell task on the protocol client, as specified in section [3.7](#).

The **stdout** message is specified in section [2.2.26](#).

3.2.5.9 Task-info Message

The protocol server MUST ignore this message. The content MUST NOT be parsed. This message is specified in section [2.2.27](#).

3.2.6 Timer Events

3.2.6.1 Heartbeat Timeout

The heartbeat timeout occurs when a protocol client has not responded to a `request_heartbeat` message, as specified in section [2.2.19](#), which is sent each time the Heartbeat timer, specified in section [3.2.2.2](#), triggers.

When a heartbeat timeout occurs for a protocol client, the protocol server MUST reassign all currently processing tasks to a different protocol client in the system.

3.2.7 Other Local Events

None.

3.3 Protocol Client Worker Details

3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

For some messages the protocol client runs the task in a child process. Under these circumstances the protocol client MUST have available the standard input, standard output and standard error streams for the child process, as specified in [\[ISO-9899\]](#) section 7.19.1 item 3.

3.3.2 Timers

None.

3.3.3 Initialization

The protocol client MUST register itself in the system by calling **RegisterModule** as specified in [\[MS-FSCX\]](#). The name element in the **ModuleRegister** struct specified in [\[MS-FSCX\]](#) MUST be "fdmworker".

The protocol begins processing when the protocol server connects, as specified in section [3.2.3](#). The protocol client MUST send an **info** message to the protocol server, as specified in section [2.2.14](#). The content of the **info** message, the *info* parameter, is implementation dependent, and MUST be ignored by the protocol server.

3.3.4 Higher-Layer Triggered Events

3.3.4.1 Stable Storage Capacity Shortage

The protocol client MUST continuously monitor its available stable storage space for storing data. If the protocol client at any time detects that the available amount of stable storage becomes less than the current limit, the protocol client MUST abort all running tasks and send **finish** messages, as specified in section [2.2.9](#), with status **aborted** as appropriate.

The limit is specified by the protocol server in a **disk_limit** message, as specified in section [2.2.7](#). If no such message is sent by the protocol server, a limit of 2000 megabytes, that is, 2^{20} bytes MUST be assumed.

3.3.5 Message Processing Events and Sequencing Rules

Messages sent from the protocol server to the protocol client fall into three broad categories, as shown in the following table.

Message type	Section
Generic-response messages	3.3.5.1
Specific-response messages	3.3.5.2
No-response messages	3.3.5.3

The difference between these categories is the expected responses and error handling. Generic-response message processing has two responses – one initial message and one final message. Specific-response message processing has only one final response message, and no-response message processing does not send any response messages. Each message process specifies its own error handling.

3.3.5.1 Generic-response Messages

The following messages MUST be responded to with a generic response that contains a command.

The protocol client that received a generic-response message MUST first send a **slot** message, as specified in section [2.2.24](#), to the protocol server, when beginning to process the command in the generic-response message. When the command finishes processing, the protocol client sends a **finish** message, as specified in section [2.2.9](#), that contains the status of the command to the protocol server. This is shown in the sequence diagram in the following figure.

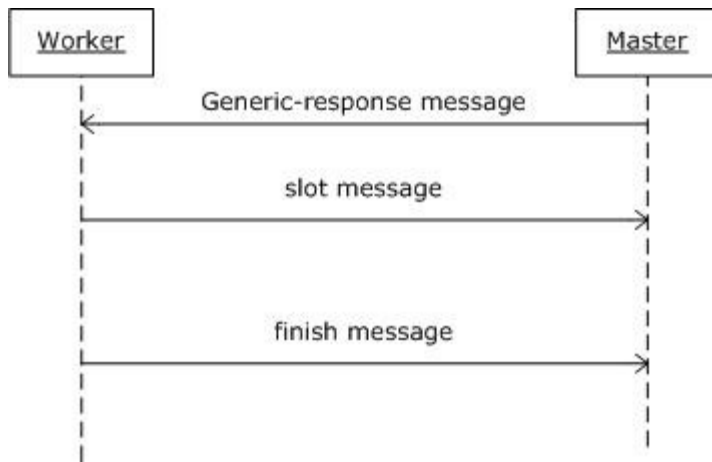


Figure 3: Handling a generic-response message

If errors occur that prevent the protocol client from processing, the protocol client MUST send an **error** message, as specified in section [2.2.8](#), to the protocol server. If an error occurred before the **slot** message was sent, the protocol client sends an **error** message. If the error occurs during command processing, the protocol client sets the *task-status* parameter of the **finish** message to the string "error" and sends it to the protocol server.

The following messages are generic-response messages:

Message	Definition	Message handling
netsink	2.2.15	Section 3.4
netsource	2.2.16	Section 3.5
remove-files	2.2.18	Section 3.3.5.1.1
shell	2.2.21	Section 3.6 and section 3.7

3.3.5.1.1 Remove-files Message

The protocol client MUST immediately remove the specified files from its file system. The files are located by their paths.

If the *directory* parameter contains the string "inputdir=" as a prefix, then it specifies a mirror directory that might contain copies of files to delete. This MUST be determined by comparing the **FileId (1)** of a file to delete to all the files in this directory. For each subsequent file path to delete, the protocol client MUST determine whether there are files in the mirror directory with matching FileId (1). In that case, the protocol client MUST ensure that the file/directory attributes of the file in the mirror directory is not affected by the removal of the file to be deleted.

3.3.5.1.2 Shell Message

This message, as specified in section [2.2.21](#), specifies that the *command* and *command-args* parameters MUST be processed in a child process shell by the protocol client. If the task cannot begin processing, the protocol client MUST respond immediately with an error message, as specified in section [2.2.8](#).

If the command can process, the protocol client MUST send a **slot** message, as specified in section [2.2.24](#). The *task-type* parameter is set to "shell", the *task-ID* parameter is set to the *task-ID* supplied in the **shell** message command, and the *task-state* parameter is set to the process identifier (PID) of the process that began processing.

3.3.5.1.2.1 Table Functions

The following five types of **shell** message *commands* operate on tables.

Map Function: The map function alters the content of a table by adding, modifying or removing records or columns. Map functions are processed by applying the function iteratively to each record in the table.

Merge Function: The merge function creates a new table from two or more input tables. For each table the column to compare is specified. A merge function is specified that is applied to each set of records.

For a specified key, at most one input table can contain duplicates. If none of the tables contain duplicate keys, the merge function is applied once for each unique key-value pair. If a table has duplicates for a key, the merge function is applied once for each duplicate.

A merge can be done in reduce mode, in which the function is applied only once for all records of a duplicate key.

Reduce Function: The reduce function alters the contents of a table by adding, modifying or removing records or columns. It applies the function iteratively to all sets of records that contain the specified key.

Sort Function: When a table is sorted on a specified key, records occur in an order specified by the value a record has for that column.

Tables can be sorted by multiple keys, in which a set of keys for two records are compared one by one until all keys have been compared, or until a mismatch has been found.

Tables can be sorted in ascending or descending order. The default sorting is ascending; descending order is referred to as reverse sorting.

Sorting is performed by comparing the specified fields of two table records. If the sorting is specified as a numeric sort, then table fields are sorted as numbers. Otherwise, the default is to compare the values octet by octet.

Tables can optionally have duplicates removed during sorting, where only one instance of the duplicate is retained. Two records are duplicates if the entire set of keys is identical for both records.

With the exception of suppressing duplicates, and the fact that the order is changed, sort does not alter the contents of a table.

Split Function: A table was split on a specified column if all occurrences of the same value in a column are located within the same physical file. The split factor for the table is based on the number of files that contain the table.

Splitting a table is achieved by calculating a hash for a value, and mapping that checksum to one of the files. Unless otherwise stated, the hash MUST be the CRC32 checksum as specified in [IEEE802.3](#) section 3.2.8. The checksum MUST be interpreted as a signed 32 bit integer.

Mapping a value to a file MUST be done by calculating the checksum modulo the number of files. The modulo operator MUST retain the same sign as the divisor.

When a value in a column is mapped to a file, the entire record containing that value MUST be put in the new file.

Splitting a table does not alter the contents of a table.

3.3.5.2 Specific-response Messages

The following messages MUST produce a specific response to the protocol server. The type of the response is specified for each message.

Message	Message definition section	Message handling definition section
abort	2.2.4	3.3.5.2.1
forget	2.2.12	3.3.5.2.2
poll-task	2.2.17	3.3.5.2.3
request_heartbeat	2.2.19	3.3.5.2.4
simple_command	2.2.22	3.3.5.2.5

3.3.5.2.1 Receiving Abort Message

The protocol client MUST terminate the task specified in the *task-ID* parameter. When the task is terminated the protocol client MUST send a **finish** message, as specified in section [3.2.5.2](#). The *task-status* parameter of the **finish** message MUST contain the string "aborted", as specified in section [2.2.4](#).

3.3.5.2.2 Receiving Forget Message

This message, as specified in section [2.2.12](#), is sent by the protocol server to indicate that it has acknowledged the message from the protocol client that specifies that a task was finished. Upon receiving this message, the protocol client MUST release any resources pertaining to the specified task.

3.3.5.2.3 Receiving Poll-task Message

The protocol server sends this message to request progress status from tasks running on protocol clients. Upon receiving a **poll-task** message, as specified in section [2.2.17](#), the protocol client MUST send a **task-info** message, as specified in [2.2.27](#), with data from the task specified in the *task-ID* parameter.

If the *task-ID* parameter refers to an unknown task the protocol client MUST send an **info** message, as specified in section [2.2.14](#). The parameter for the **info** message contains a human readable error message that specifies the error condition,

as specified in section [2.2.17](#).

3.3.5.2.4 Receiving Request_heartbeat Message

This message, as specified in section [2.2.19](#), is sent by the protocol server when querying status information from a protocol client process. Upon receiving a **request_heartbeat** message, the protocol client MUST send a **heartbeat** message, as specified in section [2.2.13](#).

If a temporary directory parameter is present, the protocol client MUST include the **tmpsize** element in the **heartbeat_info** structure, as specified in [2.2.3.2](#).

If a *heartbeat-ID* parameter is present, the protocol client MUST include the **id** element in the **heartbeat_info** structure, as specified in section [2.2.3.2](#).

3.3.5.2.5 Receiving Simple_command Message

This message, specified in section [2.2.22](#), signifies that the protocol client MUST perform the specified action synchronously on behalf of the protocol server. The task is specified in the *command* parameter, and the parameters and their names are specified in section [2.2.22](#).

The specific actions to perform for the command parameter are specified in sections [3.3.5.2.5.1](#) to [3.3.5.2.5.7](#). Each command MUST yield a corresponding result encoded in a string.

Having performed the required actions for the command parameter, the protocol client MUST send a **simple_response** message. The *task-ID* parameter in the **simple_response** message MUST be identical to that received in the **simple_command** message. The *number-of-bytes* parameter MUST be set to the length of the encoded string result. Immediately following the sending of the **simple_response** message, the protocol client MUST send the string result in full. Finally the protocol client MUST send an ASCII line feed character (hex 0x0A). The total number of bytes transmitted following the **simple_response** message is thus one greater than the result string.

If the protocol client fails to perform the requested action it MUST reply with an **error** message, as specified in section [2.2.8](#).

3.3.5.2.5.1 create_builddirs

Upon receiving a **simple_command** message with the command **create_builddirs**, the protocol client MUST create the directories named by the *build-dir* and *input-dir* parameters within the directory specified by the *parent-dir* parameter. A directory named by the *tmp-dir* parameter MUST be created within the directory named by the *build-dir* parameter.

If any intermediate path segment does not exist, including in the *parent-dir* parameter, they MUST be created.

The string result for this message MUST be two *directory_listing* arrays, specified in section [2.2.3.4](#), concatenated together. The first MUST be a shallow listing of the *input-dir* parameter. The second MUST be a deep listing of the *build-dir* parameter. However, files that do not have their file/directory attributes set to read-only MUST be ignored.

3.3.5.2.5.2 create_builddirs_no_tmp

Upon receiving a **simple_command** message with the command **create_builddirs_no_tmp**, the protocol client MUST create the directories named by the *build-dir* and *input-dir* parameters within the directory specified by the *parent-dir*.

If any intermediate path segment does not exist, including in the *parent-dir* parameter, they MUST be created.

The handling of this message MUST be done identically to the specification for the **create_builddirs** command specified in section [3.3.5.2.5.1](#), except that there is no third directory specified by a *tmp-dir* parameter to create. This includes how the result string is produced.

3.3.5.2.5.3 find_builddirs

Upon receiving a **simple_command** message with the command **find_builddirs**, the protocol client MUST inspect the directory specified in the *parent-dir* parameter. Any subfile or subdirectory whose name contains the string specified by the *path-token* parameter MUST be included in a *file_listing* array, specified in section [2.2.3.5](#), which is written to the result string. The directory MUST NOT be traversed recursively, and that the *file_listing* entries MUST only consist of the file names themselves.

3.3.5.2.5.4 link_to_file

Upon receiving a **simple_command** message with the command **link_to_file**, the protocol client MUST create a hard link in the location specified in the *dest-file* parameter. The hard link MUST refer to the file specified in the *source-file* parameter.

The result string MUST be set to the empty string.

3.3.5.2.5.5 markfile

Upon receiving a **simple_command** message with the command **markfile**, the protocol client MUST alter the file/directory attributes of the file specified in the *file* parameter and set it to read-only.

The result string MUST be set to the number of bytes the file contains, represented as a base 10 integer.

3.3.5.2.5.6 remove

Upon receiving a **simple_command** message with the command **remove**, the protocol client MUST handle this message identically to **remove-files** message processing, specified in section [3.3.5.1.1](#), except that the protocol client MUST NOT send a **slot** message, as specified in section [2.2.24](#).

The result string MUST be set to the empty string.

3.3.5.2.5.7 touch

Upon receiving a **simple_command** message with the command **touch**, the protocol client MUST create an empty file in the location specified in the *file* parameter, if none exists.

The result string MUST be set to the empty string.

3.3.5.3 No-response Messages

The following messages MUST NOT produce a response to the protocol server.

Message	Message definition section	Message handling definition section
cd	2.2.5	3.3.5.3.1
close-stdin	2.2.6	3.3.5.3.2
disk_limit	2.2.7	3.3.5.3.3
finish-tasks	2.2.10	3.3.5.3.4
info	2.2.14	3.3.5.3.5

Message	Message definition section	Message handling definition section
setenv	2.2.20	3.3.5.3.6

3.3.5.3.1 Receiving CD Message

Upon receiving a **cd** message the protocol client MUST change its current directory to that specified in the directory parameter. All subsequent unqualified path references MUST be relative to this directory.

3.3.5.3.2 Receiving Close-stdin Message

Upon receiving a **close-stdin** message the protocol client MUST close the standard input stream file descriptor of the child process running on the protocol client that is specified in the *task-ID* parameter.

3.3.5.3.3 Receiving Disk_limit Message

Upon receiving a **disk_limit** message the protocol client MUST ensure that it has at least the amount of stable storage available specified by the *number-of-mb* parameter.

3.3.5.3.4 Receiving Finish-tasks Message

This message MUST be sent by the protocol server to indicate that the sequence of tasks has been issued by the protocol server. The protocol client MUST send a **finish-tasks-ack** message, as specified in section [2.2.11](#), when it is in an idle state.

3.3.5.3.5 Receiving Info Message

This message is used to convey informational data for logging purposes, and can contain any number of parameters. It MUST contain an informational version string, but it MUST NOT be used for versioning or capability negotiation. The message contents MUST NOT be parsed.

3.3.5.3.6 Receiving Setenv Message

The protocol client MUST set the specified variable in its environment variables, and make it available in all processes when it begins to process them. Receiving this message MUST NOT cause the protocol client to send a message. The effect of setting the environment variable is implementation dependent.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 Netsink Client Details

3.4.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the

explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The following diagram specifies the state chart for the netsink handling of a netsink-netsource transaction.

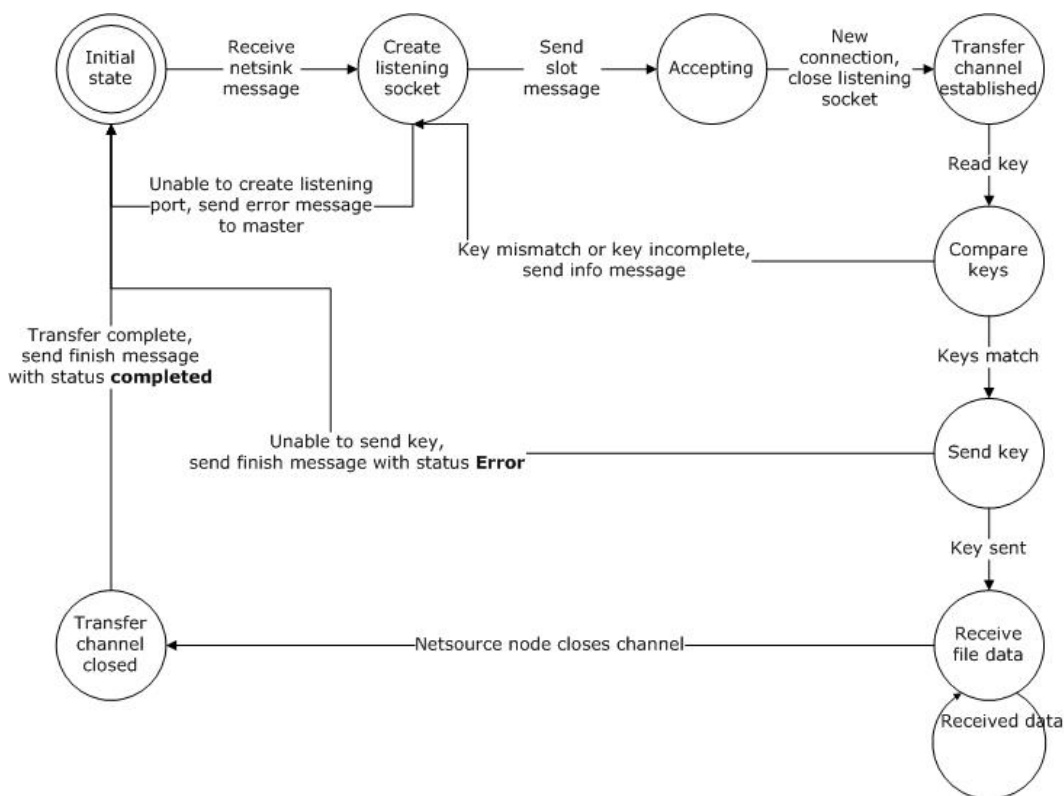


Figure 4: The netsink in a netsink-netsource transaction

3.4.2 Timers

None.

3.4.3 Initialization

The protocol server initializes the protocol by sending a **netsink** message to the protocol client.

3.4.4 Higher-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rules

Upon receiving a **netsink** message, as specified in section 2.2.15, the protocol client, or netsink, MUST bind to a TCP port that the netsource can connect to, and inform the protocol server of this port with a **slot** message, as specified in section 2.2.24, with the string "netsink" as the *task-type* parameter, and the TCP port number in a base 10 string representation as the *task-state* parameter.

If the netsink is unable to bind to a port, it MUST send an **error** message to the protocol server, as specified in section [2.2.8](#), and abort the transaction.

The netsink MUST engage in a netsink-netsource transaction as specified in the following list:

- Accepts an incoming connection from a netsource
- Verifies a key from the netsource
- Sends the key to the netsource
- Receives file data from the netsource
- Sends the **finish** message to master protocol server

Unless stated otherwise, if a failure occurs, the netsink MUST abort the transaction by sending a **finish** message, as specified in section [2.2.9](#), with the *status* parameter set to the string "error", and omit any remaining steps.

The netsink MUST wait for an incoming connection from the netsource until one is received, or until the task is terminated by the protocol server. If the netsink is unable to accept a connection, it MUST send an **error** message to the protocol server, as specified in section [2.2.8](#), and abort the transaction.

Accepting an incoming connection from a netsource protocol client:

When the netsink receives the **netsink** message, it opens a TCP port to receive connections. Then it MUST close the listening port immediately. The new connection forms a new transfer channel.

Verifying the key from the netsource:

The netsink MUST read the netsource transfer key from the transfer channel. The number of bytes to read is the same as the length of the *transfer-key* parameter, as specified in section [2.2.1.3](#). The netsink compares the transfer key from the transfer channel to the transfer key of the **netsink** message, as a byte-by-byte comparison.

If the keys match, the netsink sends the *transfer-key* parameter of the **netsink** message on the transfer channel. If the keys do not match, the netsink MUST close the transfer channel and send an **info** message that contains an error message to the protocol server, as specified in section [2.2.14](#). It MUST NOT send a **finish** message to the protocol server.

Then, the netsink restarts the transaction. It creates a new listening port and sends a message to the protocol server, the same way it receives a new identical **netsource** message from the protocol server.

Sending the key to the netsource:

The netsink MUST write the transfer key on the transfer channel for verification.

Receiving file data from the netsource:

The netsink MUST read all data sent by the netsource and write it to a file which name is specified by the *destination-path* parameter in the **netsink** message, as specified in section [2.2.15](#). The file MUST be overwritten if it already exists, and MUST be created if it does not exist.

If the *compress* parameter of the **netsink** message contains the string "compress", the netsink MUST decompress the data received from the netsource using the ZLIB algorithm, as specified in [\[RFC1950\]](#), prior to writing it to the file.

When the netsource closes the connection, the netsink MUST close the named file and send a **finish** message to the protocol server.

Sending the finish message to the master protocol server

The netsink MUST send a **finish** message the *task-status* parameter of which is specified in section [2.2.2.2](#). If the netsink-netsource transaction finishes successfully, the netsink MUST include the number of bytes received from the netsource as an auxiliary parameter in the file data.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 Netsource Client Details

3.5.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The following diagram specifies the state chart for the netsource handling of a netsink-netsource transaction.

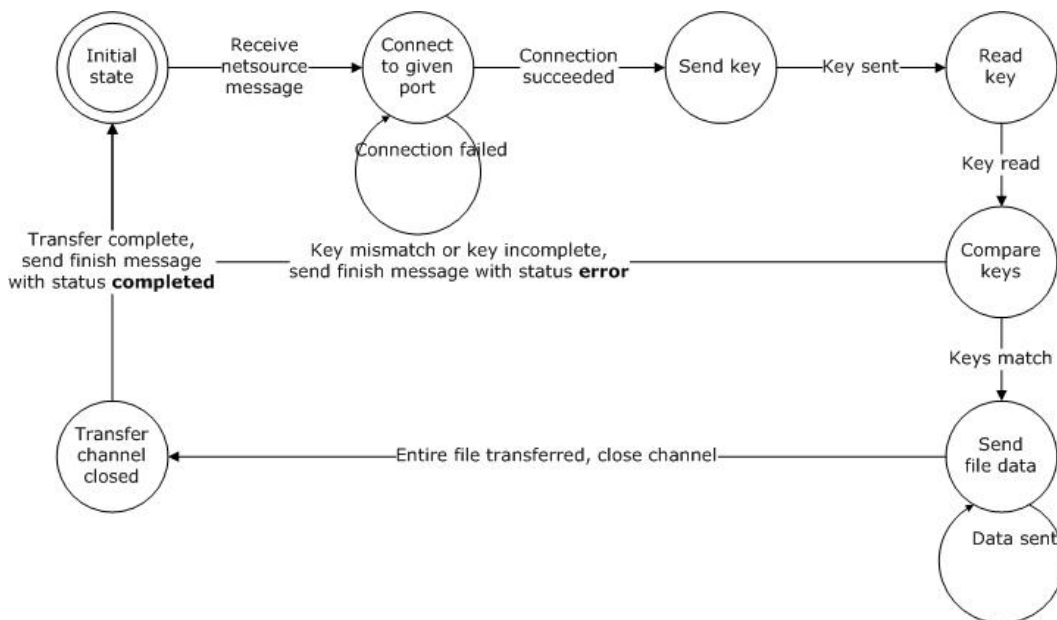


Figure 5: The netsource in a netsink-netsource transaction

3.5.2 Timers

None.

3.5.3 Initialization

The protocol server initializes the protocol by sending a **netsource** message to the protocol client.

3.5.4 Higher-Layer Triggered Events

None.

3.5.5 Message Processing Events and Sequencing Rules

Upon receiving a **netsource** message as specified in section [2.2.16](#) the protocol client, or netsource, MUST connect to the address of the netsink, as specified by the *receiver-hostname* and *receiver-port* parameters in the **netsource** message. If connecting to the netsink fails, the netsource MUST continue to attempt to establish the connection until it succeeds or the task is terminated by the protocol server.

Unless stated otherwise, if at any point there is a failure the netsource MUST abort the transaction by sending a **finish** message, as specified in section [2.2.9](#), with the *task-status* parameter set to "error", and omitting any remaining steps.

Connecting to the netsink worker

The netsource establishes a transfer channel connection to the netsink by using the *receiver-hostname* and *receiver-port* parameters in the netsource message. Then it sends the transfer key, as specified by the *transfer-key* parameter in the netsource message, on the transfer channel for verification. If the key is verified, the netsource MUST receive the same transfer key from the netsink.

The netsource MUST read the exact same number of bytes as the length of the *transfer-key* parameter received in the netsource message from the protocol server, as specified in section [2.2.16](#). The netsource MUST then perform a byte-by-byte comparison of the received netsink transfer key and the transfer key received from the protocol server in the preceding netsource message.

The transfer key verification process is the same for the netsource as the netsink, except the netsink receives a transfer key prior to sending it to the netsource.

If the transfer keys match, the netsource MUST send the entire file to the transfer channel. If the keys do not match, the netsource MUST abort the transaction immediately and close the transfer channel. It MUST then send a **finish** message with a *task-status* set to the string "error", as specified in section [2.2.2.2](#), to the protocol server.

If the netsource sends the file, and if the *compress* parameter of the **netsource** message was set to "compress", the netsource MUST compress the data using the ZLIB algorithm, as specified in [\[RFC1950\]](#), prior to writing the file to the transfer channel.

When the file has been sent, the netsource MUST close the transfer channel immediately, and send a **finish** message to the protocol server. The *status* parameter specifies the disposition of the transaction. If the netsink-netsource transaction finishes successfully, the **finish** message MUST contain the number of bytes sent to the transfer channel as an auxiliary parameter.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 Child Process Shell Cobra Client

This section specifies the **shell cobra** command. All other **shell** commands are specified in section [3.7](#).

The **shell** command functions as a virtual protocol client on a separate computer when the netsink or the netsource use it to process information.

3.6.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

If the command can process, the protocol client MUST send a **slot** message, as specified in section [2.2.24](#). The *task-type* parameter is set to "shell", the *task-ID* parameter is set to the *task-ID* supplied in the **shell** message command, as specified in section [2.2.21](#), and the *task-state* parameter is set to the process identifier (PID) of the process that has begun processing.

3.6.2 Timers

None.

3.6.3 Initialization

None.

3.6.4 Higher-Layer Triggered Events

None.

3.6.5 Message Processing Events and Sequencing Rules

The following sections specify the actions to perform upon receiving a cobra command specified in section [2.2.21.1](#).

Cobra command name	Section
merge.pyc	3.6.5.1
pythontasks.pyc	3.6.5.2
reduce.pyc	3.6.5.3
split.pyc	3.6.5.4

3.6.5.1 Receiving Merge.pyc Command

The merge.pyc message, as specified in section [2.2.21.1.1](#), contains one of 25 merge functions to apply. The merge.pyc message signifies that the protocol client MUST merge the contents of the files specified in the *mergeinputspecs* parameters using the function specified in the *merge function* parameter.

The tableoutputspec, as specified in section [2.2.3.10](#), specifies how to name the output files and how to distribute records among them. The split_factor part of the tableoutputspec determines how many output files the reduce function MUST create. The output files MUST be named according to the following pattern:

```
filename_prefix.0, filename_prefix.1, ... , filename_prefix.[split_factor -1]
```

In other words, each file MUST be named using the **filename_prefix** element concatenated with the ASCII dot character "." (hex 0x2E), and again concatenated with a unique number ranging between 0 and the value of the split_factor.

The reduce function MUST distribute records among the output files by hashing the value in the column indicated by the key part of the tableoutputspec.

The overall purpose of a merge message is specified in section [3.3.5.1.2.1](#). The protocol client MUST apply the merge function as specified by the *merge-function* parameter. Unless otherwise stated, there are two input tables to merge. The input formats of the table files are specified for each function as a *main table* and an *input table*. If there are more than two tables to merge, input tables are specified as *input table1*, *input table2* and so on. The output format is specified as an *output table*.

If the reduce parameter is present, the merge function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Sections [3.6.5.1.1](#) through [3.6.5.1.23](#) specify the actions to perform for each specific function.

3.6.5.1.1 Receiving Spmakefileutils.merge_with_global_count Message

This message specifies that the spmakefileutils.merge_with_global_count function MUST be applied on the named input files.

Main table: MUST be of type local_querycnt_by_query.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.7.

Input table: MUST be of type global_querycnt_by_query.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.4.

Output table: MUST be of type cid_by_cid_with_counts_and_query.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.2.

The function MUST emit one record per record in the *main table*, if a corresponding record exists in the input table. Before emitting records, the records for each key field MUST be sorted in descending order by comparing the LCOUNT field.

The values of the output records are specified in the following table.

Output values	How obtained
CID	The CID value of the main table record.
LCOUNT	The LCOUNT value of the main table record
GCOUNT	The GCOUNT value of the input table record.
PLACE	The index of the current record in the list of records sorted descending on the LCOUNT value, beginning at 1. The first record for the specified LCOUNT MUST use the true index as its index value, but all following records of the same LCOUNT value MUST inherit this value.
NUMURLS	The number of records for the specified LCOUNT value.
QUERY	The QUERY value of the main table record.

3.6.5.1.2 Receiving Spmakefileutils.merge_with_query Message

This message specifies that the `spmakefileutils.merge_with_query` function MUST be applied on the named input files.

Main table: MUST be of type `semi_local_querycnt_by_queryid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.13.

Input table: MUST be of type `<gen>.queries_by_queryid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.26.

Output table: MUST be of type `semi_local_querycnt_pre_token.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.16.

The function MUST emit one record per record in the *main table*, if a corresponding record exists in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URLID	The URLID field of the main table record.
LCOUNT	The LCOUNT field of the main table record.
LCID	The LCID field of the main table record.
QUERY	The QUERY field of the input table record.

3.6.5.1.3 Receiving Spmakefileutils.merge_with_url Message

This message specifies that the `spmakefileutils.merge_with_url` function MUST be applied on the named input files.

Main table: MUST be of type `local_querycnt_by_urlid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.11.

Input table: MUST be of type `<gen>.urls_by_urlid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.29.

Output table: MUST be of type `local_querycnt_by_url.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.9.

The function MUST emit one record per record in the *main table*, if a corresponding record exists in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URL	The URL field of the corresponding input table record, normalized as specified in [RFC3986] section 6.
LCOUNT	The LCOUNT field of the main table record.
QUERY	The QUERY field of the main table record.

3.6.5.1.4 Receiving Spmakefileutils.normalize_url Message

This message specifies that the `spmakefileutils.normalize_url` merge function MUST be applied on the named input files.

Main table: MUST be of type `local_querycnt_by_url.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.9.

Input table: MUST be of type `uris_by_member.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.18.

Output table: MUST be of type `local_querycnt_by_cid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.5.

The function MUST emit one record per record in the *main table*.

The values of the output records are specified in the following table.

Output values	How obtained
CID	If an input table record exists for the key, the CID field from the input table record, otherwise, the URL field of the main table record.
LCOUNT	The LCOUNT field of the main table record.
QUERY	The QUERY field of the main table record.

3.6.5.1.5 Receiving Wamakefileutils.compute_linkscore_divide_by_linkcount Message

This message specifies that the `wamakefileutils.compute_linkscore_divide_by_linkcount` merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type `rank_links_by_src`, as specified in [\[MS-FSWADF\]](#) section 2.4.1.

Input table: MUST be of type `rank_by_uri`, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

Output table: MUST be of type `linkscore_by_dst`, as specified in [\[MS-FSWADF\]](#) section 2.4.3.

The function MUST emit one record for all records in the main table, if there exists a corresponding key in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	The FROM field of the main table record.
TO	The TO field of the main table record.
RANK	MUST be a quality score for the TO field of the main table record. The exact calculation is implementation dependent. One possibility is to use the RANK field from the input table record.

3.6.5.1.6 Receiving Wamakefileutils.compact_links_and_merge_fromsite Message

This message specifies that the wamakefileutils.compact_links_and_merge_fromsite merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Input table: MUST be of type sitemap, as specified in [\[MS-FSWADF\]](#) section 2.2.5.

Output table: MUST be of type links_by_to_raw, as specified in [\[MS-FSWADF\]](#) section 2.3.2.

Between one and *N* records MUST be emitted for each key in the main table. *N* is specified for each key as the number of initial records that have the same TIMESTAMP field.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	FROM field of the main table record.
SITE	SITE field from the input table, or FROM field of the main table record.
TO	TO field from the main table record.
INTRA	INTRA field of the main table record.
ANCHORTEXT	ANCHORTEXT field of the main table record.

3.6.5.1.7 Receiving Wamakefileutils.cut_fromuris_and_reverse_urihash Message

This message specifies that the wamakefileutils.cut_fromuris_and_reverse_urihash merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type urimap, as specified in [\[MS-FSWADF\]](#) section 2.2.7.

Input table 1: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Input table 2: MUST be of type no_links, as specified in [\[MS-FSWADF\]](#) section 2.2.4.

Output table: MUST be of type pupdateuris_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.6.2.

The function MUST emit one record for all records in the main table, if there exists a corresponding key in either input table 1 or input table 2.

The values of the output records are specified in the following table.

Output values	How obtained
URL	The URL field from the main table record.

3.6.5.1.8 Receiving Wamakefileutils.extend_link_with_freqs Message

This message specifies that the wamakefileutils.extend_link_with_freqs merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type links_norm_with_fromrank_by_anchor, as specified in [\[MS-FSWADF\]](#) section 2.4.4.

Input table: MUST be of type anchor_freqs_by_anchor, as specified in [\[MS-FSWADF\]](#) section 2.4.5.

Output table: MUST be of type links_with_freqs_by_to, as specified in [\[MS-FSWADF\]](#) section 2.4.6.

The function MUST emit one record for all records in the main table, if there exists a corresponding key in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
RANK	RANK field from main table record.
AFREQ	AFREQ field from input table record.
ARANK	ARANK field from input table record.
TO	TO field from main table record.
ANCHORTEXT	ANCHORTEXT field from main table record.

3.6.5.1.9 Receiving Wamakefileutils.filter_intra_and_merge_tosite Message

This message specifies that the wamakefileutils.filter_intra_and_merge_tosite merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type links_by_raw_to, as specified in [\[MS-FSWADF\]](#) section 2.3.2 .

Input table: MUST be of type sitemap, as specified in [\[MS-FSWADF\]](#) section 2.2.5.

Output table: MUST be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

The function MUST output one record per record in the main table, if one of the following conditions is true.

1. If there exists a corresponding record in the input table, and the SITE field of the records from the main table does not match the SITE field of the record from the input table.
2. If there does not exist a corresponding record in the input table, and the INTRA field of the main table record is **0** (ASCII 0x30).

The values of the output records are specified in the following table.

Output values	How obtained
FROM	FROM field from main table record.
TO	TO field from main table record.
ANCHORTEXT	ANCHORTEXT field from main table record.

3.6.5.1.10 Receiving Wamakefileutils.logcompact_eqrepr Message

This message specifies that the wamakefileutils.logcompact_eqrepr merge function **MUST** be applied on the named input files. The function **MUST** be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: **MUST** be of type eqrepr, as specified in [\[MS-FSWADF\]](#) section 2.2.2.

Input table: **MUST** be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

Output table: **MUST** be of the type eqrepr, as specified in [\[MS-FSWADF\]](#) section 2.2.2.

The function **MUST** output one record per record in the main table, unless one of the following conditions is true:

1. If the EQREPR field is set to the NULL byte (hex 0x00) for a record
2. If the TIMESTAMP field is smaller than the TIMESTAMP field of the first record with the same key.
3. If there exists a corresponding record in the input table with a higher TIMESTAMP field.

The records for a key **MUST** be processed in the order they appear. If one of the previous conditions is true, all remaining records for the key **MUST** be discarded.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	URLHASH field from the main table record.
URL	URL field from the main table record.
TIMESTAMP	TIMESTAMP field from the main table record.
EQREPR	EQREPR field from the main table record.

3.6.5.1.11 Receiving Wamakefileutils.filter_sitemap Message

This message specifies that the wamakefileutils.filter_sitemap function **MUST** be applied on the named input files.

Main table: MUST be of type sitemap, as specified in [\[MS-FSWADF\]](#) section 2.2.5.

Input table1: MUST be of type urimap, as specified in [\[MS-FSWADF\]](#) section 2.2.7.

Input table2: MUST be of type no_links, as specified in [\[MS-FSWADF\]](#) section 2.2.4.

Output table: MUST be of type sitemap, as specified in [\[MS-FSWADF\]](#) section 2.2.5.

The function MUST emit one record per record in the *main table*, if there exists a record in *input table1* or *input table2* for the same key.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	URLHASH field from the main table record.
SITE	SITE field from the main table record.

3.6.5.1.12 Receiving Wamakefileutils.logcompact_links Message

This message specifies that the wamakefileutils.logcompact_links merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Input table: MUST be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

Output table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

The function MUST emit one record per record in the main table, unless one of the following conditions is true:

1. If there exists a corresponding record in the input table with a higher TIMESTAMP field.
2. If the TIMESTAMP field is smaller than the TIMESTAMP field of the first record with the same key.

The records for a key MUST be processed in the order they appear. If one of the previous conditions is true, all remaining records for the key MUST be discarded.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	FROM field from the main table record.
TO	TO field from the main table record.
INTRA	INTRA field from the main table record.
TIMESTAMP	TIMESTAMP field from the main table record.
ANCHORTEXT	ANCHORTEXT field from the main table record.

3.6.5.1.13 Receiving Wamakefileutils.logcompact_no_links Message

This message specifies that the wamakefileutils.logcompact_no_links merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type no_links, as specified in [\[MS-FSWADF\]](#) section 2.2.4.

Input table: MUST be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

Output table: MUST be of type no_links, as specified in [\[MS-FSWADF\]](#) section 2.2.4.

The function MUST emit one record per record in the main table, unless one of the following conditions is true:

1. If there exists a corresponding record in the input table with a higher TIMESTAMP field.
2. If the TIMESTAMP field is smaller than the TIMESTAMP field of the first record with the same key.

If one of the previous conditions is true, all remaining records for the key MUST be discarded.

The values of the output records are specified in the following table.

Output values	How obtained
URL	URL field from the main table record.
URLHASH	URLHASH field from the main table record.
TIMESTAMP	TIMESTAMP field from the main table record.

3.6.5.1.14 Receiving Wamakefileutils.logcompact_urieq Message

This message specifies that the wamakefileutils.logcompact_urieq merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type urieq, as specified in [\[MS-FSWADF\]](#) section 2.2.6.

Input table: MUST be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

Output table: MUST be of type urieq, as specified in [\[MS-FSWADF\]](#) section 2.2.6.

The function MUST emit one record per record in the main table, unless one of the following conditions is true:

1. If there exists a corresponding record in the input table with a higher TIMESTAMP field.
2. If the URLHASH and MEMBER fields are identical for a record.
3. If the TIMESTAMP field is smaller than the TIMESTAMP field of the first record with the same key.

The records for a key MUST be processed in the order they appear. If one of the previous conditions is true, all remaining records for the key MUST be discarded.

The values of the output records are specified in the following table.

Output values	How obtained
CLASS	CLASS field from the main table record.
MEMBER	MEMBER field from the main table record.
TIMESTAMP	TIMESTAMP field from the main table record.

3.6.5.1.15 Receiving Wamakefileutils.make_new_urihashmap Message

This message specifies that the wamakefileutils.make_new_urihashmap merge function **MUST** be applied on the named input files.

Main table: **MUST** be of type urimap, as specified in [\[MS-FSWADF\]](#) section 2.2.7.

Input table: **MUST** be of type urihash, as specified in [\[MS-FSWADF\]](#) section 2.3.5.

Output table: **MUST** be of type urimap, as specified in [\[MS-FSWADF\]](#) section 2.2.7.

The function **MUST** emit one record per record in the main table, if a corresponding record exists in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URL	URL field from the main table record.
URLHASH	URLHASH field from the main table record.

3.6.5.1.16 Receiving Wamakefileutils.merge_in_rank Message

This message specifies that the wamakefileutils.merge_in_rank merge function **MUST** be applied on the named input files. The function **MUST** be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: **MUST** be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

Input table: **MUST** be of type rank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

Output table: **MUST** be of type links_norm_with_fromrank_by_anchor, as specified in [\[MS-FSWADF\]](#) section 2.4.4.

The function **MUST** emit one record per record in the main table.

The values of the output records are specified in the following table.

Output values	How obtained
RANK	RANK field of corresponding record in the input table, if it exists; otherwise, the field 0 (ASCII 0x30).
FROM	FROM field from the main table record.
TO	TO field from the main table record.

Output values	How obtained
ANCHORTEXT	ANCHORTEXT field from the main table record.

3.6.5.1.17 Receiving Wamakefileutils.merge_repr Message

This message specifies that the wamakefileutils.merge_repr merge function MUST be applied on the named input files.

Main table: MUST be of type anchor_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.11.

Input table: MUST be of type eqrepr_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.3.4.

Output table: MUST be of type anchor_by_uri_with_repr, as specified in [\[MS-FSWADF\]](#) section 2.4.12.

Output MUST be created by emitting one record per record in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
EQREPR	EQREPR from input table record if one exists, the byte 0x00 otherwise.
SITE-RANK	SITE-RANK field from main table record.
TO-RANK	TO-RANK field from main table record.
LAFREQ	LAFREQ field from main table record.
LARANK	LARANK field from main table record.
AFREQ	AFREQ field from main table record.
ARANK	ARANK field from main table record.
SITE-OR-TO-URL	SITE-OR-TO-URL field from main table record.
ANCHORTEXT	ANCHORTEXT field from main table record.

3.6.5.1.18 Receiving Wamakefileutils.merge_siterank Message

This message specifies that the wamakefileutils.merge_siterank merge function MUST be applied on the named input files.

Main table: MUST be of type anchor_by_to, as specified in [\[MS-FSWADF\]](#) section 2.4.8.

Input table: MUST be of type siterank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.10.

Output table: MUST be of type anchor_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.11.

Output MUST be created by emitting one record per record in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
SITE-RANK	SITE-RANK field from input table record.
TO-RANK	TO-RANK field from main table record, or 0 (ASCII 0x30) if no main table record exists.
LAFREQ	LAFREQ field from main table record, or 0 (ASCII 0x30) if no main table record exists.
LARANK	LARANK field from main table record, or 0 (ASCII 0x30) if no main table record exists.
AFREQ	AFREQ field from main table record, or 0 (ASCII 0x30) if no main table record exists.
ARANK	ARANK field from main table record, or 0 (ASCII 0x30) if no main table record exists.
SITE-OR-TO-URL	URL field from main table record, or SITE-OR-TO-URL field from input table record if no main table record exists.
ANCHORTEXT	ANCHORTEXT field from main table record, or "-" (ASCII 0x2D) if no main table record exists.

3.6.5.1.19 Receiving Wamakefileutils.normalize_link Message

This message specifies that the wamakefileutils.normalize_link merge function **MUST** be applied on the named input files.

Main table: **MUST** be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

Input table: **MUST** be of type urieq_by_class, as specified in [\[MS-FSWADF\]](#) section 2.3.3.

Output table: **MUST** be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

The function **MUST** discard all records from the input table.

Output **MUST** be created by emitting the FROM and ANCHORTEXT fields from the main table records. If there exists an input table record for a main table record, the CLASS field from that record **MUST** be emitted as the TO field; otherwise, the main table record TO field **MUST** be emitted as the TO field.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	REPR from input table record if one exists, the byte 0x00 otherwise.
TO	CLASS field from the input table record, or TO field from the main table record.
ANCHORTEXT	ANCHORTEXT field from main table record.

3.6.5.1.20 Receiving Wamakefileutils.parse_site Message

This message specifies that the wamakefileutils.parse_site merge function **MUST** be applied on the named input file.

Main table: **MUST** be of type anchor_by_to, as specified in [\[MS-FSWADF\]](#) section 2.4.8.

Input table: **MUST** be of type eqrepr_by_uri as specified in [\[MS-FSWADF\]](#) section 2.3.4.

Output table: **MUST** be of type rank_by_site, as specified in [\[MS-FSWADF\]](#) section 2.4.9.

The function MUST emit one record per unique (URL, ANCHORTEXT) combination in the main table.

The values of the output records are specified in the following table.

Output values	How obtained
SITE	MUST be the string "site://" followed by the value of the SITE field, unless that field contains the value \0xc7\0x82. If it does, compute the following value to replace the SITE field: If there is no matching key in the input table, the value MUST be the host part of the URL field, as specified in [RFC3986] section 3.2.2: Otherwise, if there is a matching key in the input table, if the URL value of the input table begins with the string "ssic://", the value MUST be the host part of the EQREPR field in the input table. If the URL value does not begin with the string "ssic://", the value MUST be the host part of the URL field of the input table.
TO-RANK	MUST be the TO-RANK field of the main table record.
URL	MUST be the URL field of the main table record.

3.6.5.1.21 Receiving Wamakefileutils.reduce_anchor_and_add_rank Message

This message specifies that the wamakefileutils.reduce_anchor_and_add_rank merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type links_with_freqs_by_to, as specified in [\[MS-FSWADF\]](#) section 2.4.6.

Input table: MUST be of type rank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

Output table: MUST be of type uri_anchors_by_urihash, as specified in [\[MS-FSWADF\]](#) section 2.4.7.

For each unique TO, ANCHORTEXT combination occurring in the main table, the function MUST emit one record. For each such TO, ANCHORTEXT combination the number of records and the sum of all RANK fields MUST be recorded.

The values of the output records are specified in the following table.

Output values	How obtained
TO-RANK	0 (ASCII 0x30) if there does not exist a corresponding input table record for a key; otherwise, the TO-RANK field of the corresponding input table record.
LAFREQ	The recorded number of records for this FROM, TO combination.
LARANK	The recorded sum of all RANK fields.
AFREQ	The AFREQ field from the last record of this FROM, TO combination.
ARANK	The ARANK field from the last record of this FROM, TO combination.
TO	The TO field from the last record of this FROM, TO combination.
ANCHORTEXT	The ANCHORTEXT field from the last record of this FROM, TO combination.

3.6.5.1.22 Receiving Wamakefileutils.reverse_urihash Message

This message specifies that the wamakefileutils.reverse_urihash function MUST be applied on the named input files.

Main table: MUST be of type uri_anchors_by_urihash, as specified in [\[MS-FSWADF\]](#) section 2.4.7.

Input table1: MUST be of type urimap, as specified in [\[MS-FSWADF\]](#) section 2.2.7.

Input table2: MUST be of type sitemap, as specified in [\[MS-FSWADF\]](#) section 2.2.5.

Output table: MUST be of type anchor_by_to, as specified in [\[MS-FSWADF\]](#) section 2.4.8.

The function MUST emit one record per record in the main table, if a corresponding record exists in the *input table1*.

The values of the output records are specified in the following table.

Output values	How obtained
TO-RANK	The TO-RANK field from the main table record.
LAFREQ	The LAFREQ field from the main table record.
LARANK	The LARANK field from the main table record.
AFREQ	The AFREQ field from the main table record.
ARANK	The ARANK field from the main table record.
URL	The URL field of the corresponding input table 1 record.
SITE	The SITE field of the corresponding input table 2 record, if one exists. Otherwise the two-byte value 0xC7 0x82 MUST be used.
ANCHORTEXT	The ANCHORTEXT field from the main table record.

3.6.5.1.23 Receiving Wamakefileutils.sum_linkscore_and_new_static_rank Message

This message specifies that the wamakefileutils.sum_linkscore_and_new_static_rank merge function MUST be applied on the named input files. The function MUST be applied in reduce mode, as specified in section [3.3.5.1.2.1](#).

Main table: MUST be of type rank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

Input table: MUST be of type linkscore_by_dst, as specified in [\[MS-FSWADF\]](#) section 2.4.3.

Output table: MUST be of type rank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

The function MUST emit one record for all records in the main table.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	The URLHASH field of the main table record.
RANK	If there are records in the input table for the specified key, a sum of all input RANK fields MUST be computed. The new RANK field is then computed by adding the RANK field from the main table record.

3.6.5.2 Receiving pythontasks.pyc Command

The pythontasks.pyc message, specified in section [2.2.21.1.2](#), contains one of eight commands. The commands and their parameters are specified in section [2.2.21.1.2.1](#). The handling rules for each command are specified in sections [3.6.5.2.1](#) to [3.6.5.2.8](#).

3.6.5.2.1 Receiving _local_copy_func Message

The protocol client MUST create a file named by the *destination-path* parameter and write the contents of the file named by the *source-path* parameter to that file. If the *permissions* parameter is not set to the string "None", the destination file MUST have the permission bits in the file/directory attributes set accordingly.

3.6.5.2.2 Receiving _local_touch_func Message

The protocol client MUST create the file named by the *file-path* parameter if it does not exist. If the *permissions* parameter is not set to the string "None", the *file-path* MUST have the permission bits in the file/directory attributes set accordingly.

3.6.5.2.3 Receiving Spmakefileutils.file_finder Message

The protocol client MUST obtain a listing of the contents of the directory specified in the *linkdump-path* parameter. If the directory is empty, the message MUST NOT be processed further.

The protocol client MUST evaluate each file in the *filenames* parameter. For each file, an output file which name is a concatenation of the *output-prefix* parameter, the file name itself and the string ".0" MUST be created in the current directory.

For each file in the *filenames* parameter, its name MUST be compared to each file in the *linkdump-path* directory to determine whether they are target files as specified in section [2.2.3.11](#). A file in the *linkdump-path* directory MUST be a target file if one of the following is true:

- The collection name in the file matches the *collectionname* parameter and the name matches the file name being evaluated.
- The previously mentioned condition and the prefix matches the string "taken".

There can be several target files for each file name in the *filenames* parameter. If there are no target files, the output file created MUST remain empty, and the protocol client processes the next file in the *filenames* parameter.

If there are target files, the protocol client MUST write a concatenation of each target file found to the output file. Then, the protocol client processes the next file in the *filenames* parameter.

3.6.5.2.4 Receiving Wamakefileutils.clean_view_build_dir Message

The protocol client MUST obtain a listing of the contents of the directories named by the *build-path* and *input-path* parameters. For all files in the *build-path* directory, the protocol client MUST verify whether the file name begins with any of the strings in the *prefixes* parameters, or if the name is identical to any of the files in the *input-path* directory.

If a file name begins with one of the strings from the *prefixes* parameter the file MUST NOT be processed further.

All other files MUST be deleted.

If a file has the same name as any file in the *input_path* directory, the protocol client MUST assert that the corresponding file in *input_path* has its file/directory attributes set to read-only.

3.6.5.2.5 Receiving Wamakefileutils.compare_links_and_nolinks Message

The protocol client MUST assemble four file names based on the *index* and *prefix* parameters, as shown in the following ABNF.

```
links-input = prefix "links_by_from_tmp." index
nolinks-input = prefix "no_links_by_urihash_tmp." index
links-output = prefix "links_by_from." index
nolinks-output = prefix "no_links_by_urihash." index
prefix = token ;Specified in section 2.2.
index = token ;Specified in section 2.2.
```

The *prefix* and *index* variables are initialized to the values of the respective parameters of this message. The *split-factor* parameter is not in use and MUST be ignored.

The *link-input* and *links-output* MUST be files in the links format specified in [\[MS-FSWADF\]](#) section 2.2.3.

The *nolinks-input* and *nolinks-output* MUST be files in the no_links format specified in [\[MS-FSWADF\]](#) section 2.2.4.

The protocol client MUST write records from the *links-input* file to the *links-output* file, and correspondingly from the *nolinks-input* file to the *nolinks-output* file. The records MUST be written to the output files in the order they are read from the input files.

If there exists FROM fields in the *links-input* equal to URLHASH fields from *nolinks-input*, the record with the smallest TIMESTAMP field MUST be discarded. If the TIMESTAMP field is identical, the record from the *links-input* file MUST be discarded.

3.6.5.2.6 Receiving Wamakefileutils.file_finder Message

The protocol client MUST process this message in the same manner as receiving the *spmakefileutils.file_finder* message, as specified in section [3.6.5.2.3](#).

3.6.5.2.7 Receiving Wamakefileutils.remove_files_in_filter Message

The protocol client MUST obtain a listing of the contents of the directory specified in the *dir-path* parameter.

For all files in the *dir-path* directory, the protocol client MUST evaluate whether the file name begins with any value in the *prefixes* parameters. In that case, the protocol client MUST delete the file.

3.6.5.2.8 Receiving Wamakefileutils.remove_files_not_in_filter Message

The protocol client MUST obtain a listing of the contents of the directory specified in the *dir-path* parameter.

For all files in the *dir-path* directory, the protocol client MUST evaluate whether the file name begins with any value in the *prefixes* parameters. If it does not use any *prefixes* parameter value, the protocol client MUST delete the file.

3.6.5.3 Receiving Reduce.pyc Command

The reduce.pyc message, specified in section [2.2.21.1.3](#), contains one of 25 reduce functions to apply. The input of the function is specified in the *filename* part of the *inputspec* parameter.

The tableoutputspec in section [2.2.3.10](#) specifies how to name the output files and how to distribute records among them. The *split_factor* part of the tableoutputspec determines how many output files the reduce function MUST create. The output files must be named according to the following pattern:

```
filename_prefix.0, filename_prefix.1, ... , filename_prefix.[split_factor -1]
```

The reduce function MUST distribute records among the output files by hashing the value in the column indicated by the key part of the tableoutputspec.

The overall purpose of a reduce.pyc message is specified in section [3.3.5.1.2.1](#). The protocol client MUST apply the function specified in the *reducefunction* parameter. If no function is specified, a default function MUST be applied. The input format of the table files is specified for each function as an *input table*, whereas the output format is specified as an *output table*.

Sections [3.6.5.3.1](#) to [3.6.5.3.25](#) specify the actions to perform for each specific function.

3.6.5.3.1 Receiving Spmakefileutils.compact_uris Message

This message specifies that the spmakefileutils.compact_uris reduce function MUST be applied on the named input files.

Input table: MUST be of type <gen>.<col>.uris_by_contentid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.32.

Output table: MUST be of the type uris_by_member.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.18.

The function MUST emit one record per record in the input table, except for records with a **TIMESTAMP** value that is different from the first record with the same key, and whose **OP** value is different than the literals **URLSCHANG**E or **ADD**.

The values of the output records are specified in the following table.

Output values	How obtained
CID	The CID field of the mail table record.
MEMBER	The MEMBER field of the mail table record.

3.6.5.3.2 Receiving Spmakefileutils.local_count_queries Message

This message specifies that the spmakefileutils.local_count_queries reduce function MUST be applied on the named input files.

Input table: MUST be of type <date>.clicks_by_urlid_and_queryid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.37.

Output table: MUST be of the type <date>.local_querycnt_by_queryid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.40.

The function MUST emit one record per unique QUERYID, LCID combination. For each unique combination, the number of records MUST be counted.

The values of the output records are specified in the following table.

Output values	How obtained
URLID	The URLID field of the input table record.
QUERYID	The QUERYID concatenated with the LCID field of the input table record.
LCID	The LCID field of the input table record.
LCOUNT	The counted number of records for this QUERYID, LCID combination.

3.6.5.3.3 Receiving Spmakefileutils.filter_old_queries Message

This message specifies that the spmakefileutils.filter_old_queries reduce function MUST be applied on the named input files.

Input table: MUST be of type <gen>.queries_by_queryid_all.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.27.

Output table: MUST be of the type <gen>.queries_by_queryid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.26.

This function uses regular expressions, as specified in [\[MC-RegEx\]](#). For clarity, the regular expressions are specified in ABNF.

If the QUERY field matches the following SCOPERULE regular expression, the matching text in the QUERY field MUST be removed before further processing.

```
SCOPERULE = SP "scope:" DQUOTE ".*" DQUOTE "$"
```

The function MUST emit one record per record in the input table, except in the following cases:

1. If the day specified in the DATE field is smaller than the date specified in the *fromdate* parameter. This parameter is specified with the funcvars option, as specified in section [2.2.21.1.3](#). The *fromdate* parameter is an entry in the funcvars option dictionary where the key is the string "fromdate" and the value is a string that specifies a date in the format "YYYY-MM-DD".
2. If the QUERY field matches any of the following regular expressions. Matching MUST be case insensitive.

```
PROPRULE = "\s+:\s+"
```



```

PROPEQRULE   = "\S+=\S+"
PROPGTRULE   = "\S+>\S+"
PROPLTRULE   = "\S+<\S+"
ANDRULE      = SP "and" SP
ORRULE       = SP "or" SP
NOTRULE      = SP "not" SP
NOTBEGRULE   = "^not" SP
MINTERM      = SP "-\S+"
MINBEGTERM   = "^-\S+"
NEARRULE     = SP "near" SP
WORDSRULE    = SP "words\(\S+"
WORDSBEGRULE = "^words\(\S+"

```

The values of the output records are specified in the following table.

Output values	How obtained
QUERYID	The QUERYID field of the input table record.
DATE	The DATE field of the input table record.
QUERY	The QUERY field of the input table record, after applying the SCOPERULE regular expression.

3.6.5.3.4 Receiving Spmakefileutils.filter_old_urls Message

This message specifies that the spmakefileutils.filter_old_urls reduce function MUST be applied on the named input files.

Input table: MUST be of type <gen>.urls_by_urlid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.29.

Output table: MUST be of the type <gen>.urls_by_urlid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.29.

The function MUST emit one record per unique URLID in the input table, except if the date specified in the DATE field is smaller than the date specified in the *fromdate* parameter. If the input table contains multiple records with the same URLID, the record with the most recent date MUST be emitted. This parameter is specified with the funcvars option, as specified in section [2.2.21.1.3](#). The *fromdate* parameter is an entry in the funcvars option dictionary where the key is the string "fromdate" and the value is a string that specifies a date in the format "YYYY-MM-DD".

The values of the output records are specified in the following table.

Output values	How obtained
URLID	The URLID field of the input table record.
DATE	The DATE field of the input table record.
URL	The URL field of the input table record.

3.6.5.3.5 Receiving Spmakefileutils.full_global_count_queries Message

This message specifies that the spmakefileutils.full_global_count_queries reduce function MUST be applied on the named input files.

Input table: MUST be of type local_querycnt_by_query.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.7.

Output table: MUST be of the type global_querycnt_by_query.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.4.

The function MUST emit one record per unique QUERY value in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
GCOUNT	The sum of all LCOUNT fields of the input table records.
QUERY	The QUERY field of the input table record.

3.6.5.3.6 Receiving Spmakefileutils.full_local_count_queries Message

This message specifies that the spmakefileutils.full_local_count_queries reduce function MUST be applied on the named input files.

Input table: MUST be of type semi_local_querycnt_by_urlid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.14.

Output table: MUST be of the type local_querycnt_by_urlid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.11.

The function MUST emit one record per unique QUERY value in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URLID	The URLID field of the input table record.
LCOUNT	The sum of the LCOUNT fields for the QUERY field in the input table.
QUERY	The QUERY field of the input table record.

3.6.5.3.7 Receiving Spmakefileutils.logcompact_contentids Message

This message specifies that the spmakefileutils.logcompact_contentids reduce function MUST be applied on the named input files.

Input table: MUST be of type <gen>.<col>.uris_by_contentid_ts.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.33.

Output table: MUST be of the type <gen>.<col>.uris_by_contentid.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.32.

The function MUST emit one record per record in the input table, except for records with a **TIMESTAMP** field that is different from the first record with the same key, and whose **OP** field is different than the strings "URLSCHANG" or "ADD".

The values of the output records are specified in the following table.

Output values	How obtained
URI	The URI field of the mail table record.
TIMESTAMP	The TIMESTAMP field of the mail table record.
OP	The OP field of the mail table record.
MEMBER	The MEMBER field of the mail table record.

3.6.5.3.8 Receiving Spmakefileutils.make_uris_uniq Message

This message specifies that the `spmakefileutils.make_uris_uniq` reduce function MUST be applied on the named input file.

Input table: MUST be of type `<gen>.<col>.uris_by_contentid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.32.

Output table: MUST be of type `<gen>.<col>.unique_uris_by_uri.<sf>`, as specified in [\[MS-FSWADF\]](#) section 2.4.31.

The function MUST emit one record per unique URI field in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URI	The URI field of the input table record.

3.6.5.3.9 Receiving Spmakefileutils.reduce_localcount Message

This message specifies that the `spmakefileutils.reduce_localcount` reduce function MUST be applied on the named input files.

Input table: MUST be of type `local_querycnt_by_cid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.5.

Output table: MUST be of the type `local_querycnt_by_query.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.7.

The function MUST emit one record per unique CID, QUERY combination in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
CID	The CID field of the input table record.
LCOUNT	The sum of all LCOUNT fields for this CID, QUERY combination.

Output values	How obtained
QUERY	The QUERY field of the input table record.

3.6.5.3.10 Receiving Spmakefileutils.reduce_urls Message

This message specifies that the `spmakefileutils.reduce_urls` reduce function MUST be applied on the named input files.

Input table: MUST be of type `cid_by_cid_with_counts_and_query.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.2.

Output table: MUST be of the type `urls_on_urlhash_with_queries.sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.22.

The function MUST emit one record per unique CID key in the input table. The QUERIES field in the output table is constructed by assembling a dictionary, as specified in [\[MS-FSSPRDF\]](#) section 2.4.22.

The values of the dictionary are specified in the following table.

Dictionary value	How obtained
queries	An array of LCOUNT, GCOUNT, PLACE, NUMURLS, QUERY tuples that specifies all records associated with a key. The array MUST be serialized as specified in [MS-FSWCU] .
contentid	The CID field from the input table records.

The values of the output records are specified in the following table.

Output values	How obtained
HASH	A hash of the input table CID field. First, an MD5 digest as specified in [RFC1321] is computed. This digest is in turn interpreted as a big-endian 128-bit unsigned integer, and is converted to a base 10 decimal string.
QUERIES	The dictionary serialized as specified in [MS-FSWCU] , and then base64 encoded.

3.6.5.3.11 Receiving Spmakefileutils.tokenize_query Message

This message specifies that the `spmakefileutils.tokenize_query` reduce function MUST be applied on the named input files.

Input table: MUST be of type `semi_local_querycnt_pre_token.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.16.

Output table: MUST be of the type `semi_local_querycnt_by_urlid.<sf>`, as specified in [\[MS-FSSPRDF\]](#) section 2.4.14.

This function MUST emit one record per record in the input table.

The QUERY field MUST be a **UTF-8** octet sequence, character normalized, and tokenized as specified in [\[MS-FSIN\]](#).

The values of the output records are specified in the following table.

Output values	How obtained
URLID	The URLID field of the input table record.
LCOUNT	The LCOUNT field of the input table record.
QUERY	The tokenized QUERY field of the input table record.

3.6.5.3.12 Receiving Wamakefileutils.anchor_weight Message

This message specifies that the wamakefileutils.anchor_weight reduce function MUST be applied on the named input files.

Input table: MUST be of type links_norm_with_fromrank_by_anchor, as specified in [\[MS-FSWADF\]](#) section 2.4.4.

Output table: MUST be of the type anchor_freqs_by_anchor, as specified in [\[MS-FSWADF\]](#) section 2.4.5.

The number of records matching a key MUST be recorded. The sum of all RANK fields for a key MUST be recorded. Finally, the first ANCHORTEXT field for a key MUST be recorded.

One record MUST be emitted for each key. For each key, the number of records and the sum of all RANK fields MUST be recorded.

The values of the output records are specified in the following table.

Output values	How obtained
AFREQ	The number of records matching the key.
ARANK	The sum of all RANK for this key in the input table.
ANCHORTEXT	The ANCHORTEXT field from a record in the input table for this key.

3.6.5.3.13 Receiving Wamakefileutils.compact_eqrepr Message

This message specifies that the wamakefileutils.compact_eqrepr reduce function MUST be applied on the named input files.

Input table: MUST be of type eqrepr, as specified in [\[MS-FSWADF\]](#) section 2.2.2.

Output table: MUST be of the type eqrepr_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.3.4.

The function MUST emit one record per record in the input table, except for records with a TIMESTAMP field that is different from the first record with the same key.

The values of the output records are specified in the following table.

Output values	How obtained
URL	The URL field from the input table record.
EQREPR	The EQREPR field from the input table record.

3.6.5.3.14 Receiving Wamakefileutils.compact_links Message

This message specifies that the wamakefileutils.compact_links reduce function MUST be applied on the named input files.

Input table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Output table: MUST be of the type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

The function MUST emit one record per record in the input table, except for records with a TIMESTAMP field that is different from the first record with the same key.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	The FROM field from the input table record.
TO	The TO field from the input table record.
ANCHORTEXT	The ANCHORTEXT field from the input table record.

3.6.5.3.15 Receiving Wamakefileutils.compact_urieq Message

This message specifies that the wamakefileutils.compact_urieq reduce function MUST be applied on the named input files.

Input table: MUST be of type urieq, as specified in [\[MS-FSWADF\]](#) section 2.2.6.

Output table: MUST be of the type urieq_by_class, as specified in [\[MS-FSWADF\]](#) section 2.3.3.

The function MUST emit one record per record in the input table, except for records with a TIMESTAMP value that is different from the first record with the same key.

The values of the output records are specified in the following table.

Output values	How obtained
CLASS	The CLASS field from the input table record.
MEMBER	The MEMBER field from the input table record.

3.6.5.3.16 Receiving Wamakefileutils.cut_linkuris Message

This message specifies that the wamakefileutils.cut_linkuris reduce function MUST be applied on the named input file.

Input table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Output table: MUST be of type urihash, as specified in [\[MS-FSWADF\]](#) section 2.3.5.

The function MUST emit one record per record in the input table. In addition, having emitted all records for one key, one record for the key itself MUST be emitted.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	The TO field from the input table record when emitting for a record, or the FROM field when emitting the record containing the key.

3.6.5.3.17 Receiving Wamakefileutils.cut_urieq Message

This message specifies that the wamakefileutils.cut_urieq reduce function MUST be applied on the named input file.

Input table: MUST be of type urieq, as specified in [\[MS-FSWADF\]](#) section 2.2.6.

Output table: MUST be of type urihash, as specified in [\[MS-FSWADF\]](#) section 2.3.5.

The function MUST emit one record per record in the input table. In addition, having emitted all records for one key, one record for the key itself MUST be emitted.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	The MEMBER field from the input table record when emitting for a record, or the CLASS field when emitting the record containing the key.

3.6.5.3.18 Receiving Wamakefileutis.do_siterank Message

This message specifies that the wamakefileutis.do_siterank reduce function MUST be applied on the named input file.

Input table: MUST be of type rank_by_site, as specified in [\[MS-FSWADF\]](#) section 2.4.9.

Output table: MUST be of type siterank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.10.

The function MUST emit one record per record in the input table. In addition, having emitted all records for one key, one record for the key itself MUST be emitted.

The values of the output records are specified in the following table.

Output values	How obtained
SITE-OR-TO-URL	The URL field of the record from the input table, or the SITE field when emitting the record containing the key.
SITE-RANK	The RANK fields from the records in the input table MUST be used to create a quality score for the SITE-OR-TO-URL field. How the value is calculated is implementation dependent. One possibility is to compute an average.

3.6.5.3.19 Receiving Wamakefileutils.eqrepr_by_urihash_prep Message

This message specifies that the wamakefileutils.eqrepr_by_urihash_prep reduce function MUST be applied on the named input file.

Input table: MUST be of type eqrepr, as specified in [\[MS-FSWADF\]](#) section 2.2.2.

Output table: MUST be of type eqrepr_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.3.4.

The function MUST emit one record per record in the input table.

Output values	How obtained
URL	The URL field of the input table record.
EQREPR	The EQREPR field of the input table record.

3.6.5.3.20 Receiving Wamakefileutils.extract_links Message

This message specifies that the wamakefileutils.extract_links reduce function MUST be applied on the named input file.

Input table: MUST be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

Output table: MUST be of type rank_links_by_src, as specified in [\[MS-FSWADF\]](#) section 2.4.1.

The function MUST emit one record per record in the input table.

Output values	How obtained
FROM	The FROM field of the input table record.
TO	The TO field of the input table record.
COUNT	The number of records for the specified key.

3.6.5.3.21 Receiving Wamakefileutils.initial_rank Message

This message specifies that the wamakefileutils.initial_rank reduce function MUST be applied on the named input file.

Input table: MUST be of type rank_links_by_src, as specified in [\[MS-FSWADF\]](#) section 2.4.1.

Output table: MUST be of type rank_by_uri, as specified in [\[MS-FSWADF\]](#) section 2.4.2.

The function MUST emit one record per record in the input table. In addition, having emitted all records for one key, one record for the key itself MUST be emitted.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	The TO field from the input table record when emitting for a record, or the FROM field when emitting the record containing the key.
RANK	The string "1" (ASCII 0x31)

3.6.5.3.22 Receiving Wamakefileutils.links_by_from_prep Message

This message specifies that the wamakefileutils.links_by_from_prep reduce function MUST be applied on the named input file.

Input table: MUST be of type links, as specified in [\[MS-FSWADF\]](#) section 2.2.3.

Output table: MUST be of type links_by_to, as specified in [\[MS-FSWADF\]](#) section 2.3.1.

The function MUST emit one record per record in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
FROM	The FROM field from the input table records.
TO	The TO field from the input table records.
ANCHORTEXT	The ANCHORTEXT field from the input table records.

3.6.5.3.23 Receiving Wamakefileutils.make_unique Message

This message specifies that the wamakefileutils.make_unique reduce function MUST be applied on the named input file.

Input table: MUST be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

Output table: MUST be of type delete, as specified in [\[MS-FSWADF\]](#) section 2.2.1.

The function MUST emit one record per unique URI field in the input table.

The values of the output records are specified in the following table.

Output values	How obtained
URLHASH	The URLHASH field of the input table record.
TIMESTAMP	The TIMESTAMP field of the input table record.

3.6.5.3.24 Receiving Wamakefileutils.reduce_to Message

This message specifies that the wamakefileutils.reduce_to reduce function MUST be applied on the named input files.

Input table: MUST be of type anchor_by_uri_with_repr, as specified in [\[MS-FSWADF\]](#) section 2.4.12.

Output table: MUST be of the type anchor_info_new, as specified in [\[MS-FSWADF\]](#) section 2.4.13.

The function MUST emit one record per key in the input table. The ANCHORINFO field in the output table is constructed by assembling a dictionary as specified in [\[MS-FSWADF\]](#) section 2.4.13.

The values of the dictionary are specified in the following table.

Value	How obtained
anchors	An array of (ANCHORTEXT, LAFREQ, LARANK, AFREQ, ARANK) tuples that specify all records that are associated with a key. The array MUST be serialized as specified in [MS-FSWCU] .
contentid	The SITE-OR-TO-URL field from the input table records.
rank	The TO-RANK field from the input table records.

Value	How obtained
siterank	The SITE-RANK field from the input table records.
urieqs	The EQREPR field from the input table records, with the ASCII NULL bytes (0x00) replaced with ASCII whitespace " " (0x20), and finally all trailing and leading whitespace removed.

The values of the output records are specified in the following table.

Output values	How obtained
SITE-OR-TO-URL-HASH	A hash of the input table SITE-OR-TO-URL field, as specified in [MS-FSWADF].
ANCHORINFO	The dictionary serialized as specified in [MS-FSWADF].

3.6.5.3.25 Receiving Wamakefileutils.urieq_by_class_prep Message

This message specifies that the wamakefileutils.urieq_by_class_prep reduce function **MUST** be applied on the named input file.

Input table: **MUST** be of type urieq, as specified in [\[MS-FSWADF\]](#) section 2.2.6.

Output table: **MUST** be of type urieq_by_class, as specified in [\[MS-FSWADF\]](#) section 2.3.3.

The function **MUST** emit one record per record in the input table.

Output values	How obtained
CLASS	The CLASS field of the record in the input table.
MEMBER	The MEMBER field of the record in the input table.

3.6.5.4 Receiving Split.pyc Command

The protocol client **MUST** split the inputfile according to the outputspec, as specified in [2.2.21.1.4](#).

The outputspec specifies how to name the output files and how to distribute records among them. The split_factor part of the outputspec determines how many output files the split function **MUST** create. The output files must be named according to the following pattern:

```
filename_prefix.0, filename_prefix.1, ... , filename_prefix.[split_factor -1]
```

In other words, each file **MUST** be named using the filename_prefix element concatenated with the ASCII period character "." (hex 0x2E), and again concatenated with a unique number ranging between 0 and split_factor.

The split function **MUST** distribute records among the output files by hashing the value in the column indicated by the key part of the tableoutputspec as specified in section [2.2.3.10](#).

If the *splitfunction* parameter is present, the protocol client **MUST** interpret the key column as the string representation of a decimal integer and use this as the hash value, instead of the hash algorithm specified in section [3.3.5.1.2.1](#).

3.6.6 Timer Events

None.

3.6.7 Other Local Events

None.

3.7 Child Process Shell Client

This section specifies the **shell** command. The **shell cobra** commands are specified in section [3.6](#).

3.7.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

If the command can process, the protocol client MUST send a **slot** message, as specified in section [2.2.24](#). The *task-type* parameter is set to "shell", the *task-ID* parameter is set to the *task-ID* supplied in the **shell** message command, and the *task-state* parameter is set to the process identifier (PID) of the process that has begun processing.

The following table specifies the messages that can be processed.

Message	Definition
Create attribute	Build an index over the contents of the file specified in the <i>inputfile</i> parameter
Jsort2.exe	Sort the files in the <i>inputfiles</i> parameter as specified in the message, and write the sorted output to the <i>outputfile</i> parameter.
Make_pu_diff	Compare the files named by the <i>oldfile</i> and <i>newfile</i> parameters, keyed by the contents of the file named by the <i>mainfile</i> parameter.
Pupdateclient	Send content updates in a file specified by the <i>inputfile</i> parameter to the content collection specified by the <i>collectionname</i> parameter.
Sp_Make_pu_diff	Compare the files named by the <i>oldfile</i> and <i>newfile</i> parameters, keyed by the contents of the file named by the <i>mainfile</i> parameter.
Truncate_anchorinfo	Compare the files specified in the <i>oldfile</i> and <i>newfile</i> parameters, and write updated entries.
Truncate_clickinfo	Compare the files specified in the <i>oldfile</i> and <i>newfile</i> parameters, and write updated entries

The rules for handling messages are specified in section [3.7.5](#).

3.7.2 Timers

None.

3.7.3 Initialization

None.

3.7.4 Higher-Layer Triggered Events

None.

3.7.5 Message Processing Events and Sequencing Rules

3.7.5.1 Create_attribute_files Command

The protocol client MUST build an index of the contents in the *inputfile* parameter.

The format of the file specified by the *inputfile* parameter MUST be of type **anchor_info_new**, as specified in [\[MS-FSWADF\]](#) section 2.4.13.

The index must be created as specified in [\[MS-FSWADF\]](#) section 2.5. There are three output files that use the *outputfile* parameter as a common prefix. The file format is specified in [\[MS-FSWADF\]](#) sections [2.5.1](#) through [2.5.3](#).

The message is specified in section [2.2.21.2.1](#).

3.7.5.2 Jsort2.exe Command

The protocol client MUST sort the files in the *inputfiles* parameter as specified in the message, and write the sorted output to the *outputfile* parameter. Sorting MUST be performed as specified in section [3.3.5.1.2.1](#).

If the *size* parameter is present, this MUST be the upper limit for memory usage during sorting.

If the *unique* parameter is present, all duplicate values MUST be omitted in the output.

If the *merge* parameter is present, all input files MUST already be sorted, and the output MUST be a merge sort of these files.

If there are *sortkeyspec* parameters in the message, the protocol client MUST process the input file as a space-delimited data table, and MUST compare only the values of the selected columns in the *sortkeyspec*. The protocol client MUST apply the *sortkeyspec* parameters in the order they appeared in the message, as specified in section [2.2.21.2.2](#).

3.7.5.3 Make_pu_diff Command

The protocol client MUST compare the files named by the *oldfile* and *newfile* parameters, keyed by the contents of the file named by the *mainfile* parameter.

The format of the *oldfile* and the *newfile* MUST be as specified in [\[MS-FSWADF\]](#) section 2.4.13. The format of the *mainfile* MUST be as specified in [\[MS-FSWADF\]](#) section 2.6.2.

The results of the analysis MUST be written to a file named by the *outputfile* parameter.

The *mainfile* MUST specify all keys. The key along with the *collectionname* parameter MUST be written to the output file if one of the following conditions is true:

- If entries exist for a key, but the *oldfile* key differs from the *newfile* key.

- If the *isnew* parameter is set to 1 and entries exist for a key in both the oldfile and the newfile.
- If the key exists in the oldfile, but not in the newfile.

The format of the output file MUST be as specified in [\[MS-FSWADF\]](#) section 2.6.1. The message is specified in section [2.2.21.2.3](#).

3.7.5.4 Pupdateclient Command

The protocol client MUST send content updates in a file specified in the *inputfile* parameter to the content collection specified by the *collectionname* parameter.

The format of the input file MUST be as specified in [\[MS-FSWADF\]](#) section 2.6.1.

Each content update MUST contain only the URL of a document. They MUST be submitted to the system using the protocol specified in [\[MS-FSCF\]](#). The content updates MUST be submitted as **partial_update_operation** operations, as specified in [\[MS-FSCF\]](#) section 2.2.40.

The message is specified in section [2.2.21.2.4](#).

3.7.5.5 Sp_make_pu_diff Command

The protocol client MUST process this message the same way it processes the *make_pu_diff* message, as specified in section [3.7.5.3](#).

Upon receiving the *sp_make_pu_diff* message, specified in section [2.2.21.2.5](#), the protocol client MUST compare the files specified in the *oldfile* and *newfile* parameters, keyed by the contents of the file specified in the *mainfile* parameter.

The format of the oldfile MUST be of type <gen>.queryinfo.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.28. The format of the newfile MUST be of type urls_on_urlhash_with_queries.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.22.

The format of the mainfile MUST be of type <gen>.<col>.unique_uris_by_uri.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.31.

The results of the analysis MUST be written to a file specified by the *outputfile* parameter. The format of the output file MUST be of type <col>_feeduris.<sf>, as specified in [\[MS-FSSPRDF\]](#) section 2.4.23.

The message is specified in section [2.2.21.2.5](#).

3.7.5.6 Truncate_anchorinfo Command

The protocol client MUST compare the files specified in the *oldfile* and *newfile* parameters, keyed by the contents of the file named by the *mainfile* parameter.

The format of the oldfile and newfile MUST be as specified in [\[MS-FSWADF\]](#) section 2.4.13. The format of the mainfile MUST be as specified in [\[MS-FSWADF\]](#) section 2.6.1.

The results of the analysis MUST be written to a file named by the *outputfile* parameter. The format of the output file MUST be as specified in [\[MS-FSWADF\]](#) section 2.4.13.

The mainfile MUST specify all keys. If entries exist for a key, but differ between the oldfile and newfile, the updated entry MUST be written to the outputfile. If no new entry exists, the old entry MUST be written to the output file, and similarly if no old entry exists, the new entry MUST be written to the output file.

Entries written to the output file MUST have the **collections** attribute set according to the *collectionnames* parameter.

This message is specified in section [2.2.21.2.6](#).

3.7.5.7 Truncate_clickinfo Command

The protocol client MUST compare the files specified in the *oldfile* and *newfile* parameters, keyed by the contents of the file specified by the *mainfile* parameter.

The format of the oldfile MUST be as specified in [\[MS-FSSPRDF\]](#) section 2.4.28. The format of the newfile MUST be as specified in [\[MS-FSSPRDF\]](#) section 2.4.22. The format of the mainfile MUST be as specified in [\[MS-FSSPRDF\]](#) section 2.4.1.

The results of the analysis MUST be written to a file specified by the *outputfile* parameter. The format of the output file MUST be as specified in [\[MS-FSSPRDF\]](#) section 2.4.28.

The mainfile MUST specify all keys. If entries exist for a key, but differ between the oldfile and newfile, the updated entry MUST be written to the outputfile. If no new entry exists, the old entry MUST be written to the output file, and similarly if no old entry exists, the new entry MUST be written to the outputfile.

Entries written to the output file MUST have the **collections** attribute set according to the *collectionnames* parameter.

This message is specified in section [2.2.21.2.7](#).

3.7.6 Timer Events

None.

3.7.7 Other Local Events

None.

4 Protocol Examples

The following sections contain sample excerpts of protocol communication between protocol clients and the protocol server.

For clarity, base64 encoding has been omitted where possible, however it is explicitly stated in each case when the encoding has been omitted.

4.1 Path Name References

The following example shows a path name and its environment variable references, as specified in section [2.2.1.2](#).

The message is a **cd** command, specified in section [2.2.5](#), sent from the protocol server to the protocol client. The base64 encoding has been removed.

```
#cd $FASTSEARCH/data/webanalyzer/waworker1\
```

The parameter is a path name, and the path segment delimiter can be either forward or a backward slash. Also, the path parameter has a leading environment variable reference to the variable named *FASTSEARCH*. The protocol client then inspects its local environment and replace the environment variable reference with the value from its local environment.

4.2 cd Message

The following is an example of the message format specified in section [2.2.5](#). The base64 encoding has been removed.

```
#cd c:\\tmp\\
```

4.3 simple_command Message

The following is an example of the message format specified in section [2.2.22](#).

```
#simple_command 14
@base64:dG91Y2ggJEZBU1RTRUFSSQ0gvZGF0YS93ZWJhbmFseXplci93YXdvcmtlcjFcYnVpbGRfY29sbGVjdGlvbnNcb
WFrZV9zcC5ub19saW5rc19ieV91cmloYXNoX3RzXzExLmVuZA==
```

Decoding the base64 part results in the following message.

```
#simple_command 14 'touch
$FASTSEARCH/data/webanalyzer/waworker1\\build_collections\\make_sp.no_links_by_urihash_ts_11.
end'
```

The *task-ID* in the message is 14, and the protocol client will send a **simple_response** message as specified in section [3.3.5.2.5](#). The reply message is as follows.

```
#simple_response 14 0
```

4.4 shell_task

The following example shows the messages related to a shell message, as specified in section [3.7.5.2](#).

```
#shell #umask=54 30 cmd /C jsort2.exe -S 200M -T . -m -k2,2 2.sp.0_no_links_by_urihash_new.12
1.sp.no_links_by_urihash.12 -o 2.sp.no_links_by_urihash_ts.12
#slot shell 30 5564

#finish 30 shell completed 0
@base64:VGhlIHh5c3RlbSBjYW5ub3QgZmluZCB0aGUgZmlsZSBzcGVjaWZpZWQuDQo=

#forget 30
```

4.5 File transfer with netsink and netsource

This example shows the messages sent during a file transfer as specified in section [3.4](#) and section [3.5](#). The protocol server begins the file transfer by sending a **netsink** message to the netsink :

```
#netsink 42
@base64:YzpcdG1wXHRhbGVzXGZkbVxhdXRvdGVzdFxmZG13b3JrZXIxXGJlaWxkLTAwMFx0ZXh0X29uX3IxLjg=
@base64:LWM6XHRtcFx0YWxlc1xmZG1cYXV0b3Rlc3RcZmRtd29ya2VyMVxidWlsZC0wMDBcdGV4dF9vb19yMS44LTYyM
DQ5OTUwMjgzNDI3MDM5Nzk= nocompress
```

The number 42 in the **netsink** message is the task identifier. The same task identifier is used in the subsequent **slot**, **finish** and **forget** messages, as specified in sections [2.2.24](#), [2.2.9](#), and section [2.2.12](#) respectively, to identify the file transfer. The netsink responds with a **slot** message back to the protocol server:

```
#slot netsink 42 9002
```

The number 9002 in the **slot** message is the port number of which the netsink is listening for a connection from the netsource. The protocol server sends this port to the netsource with the following **netsource** message. After the file has been transferred to the netsink the protocol client sends a **finish** message to the protocol server and the protocol server replies with a **forget** message:

```
#finish 42 netsink completed 5034

#forget 42
```

The following messages are sent between the protocol server and the netsource during the file transfer. The port number 9002 is contained in the **netsource** message.

```
#netsource 130
@base64:YzpcdG1wXHRhbGVzXGZkbVxhdXRvdGVzdFxmZG13b3JrZXIxXGJlaWxkLTAwMFx0ZXh0X29uX3IxLjE0
8P8HB4J.ad.fast.no 9002
```

The netsource replies to the protocol server with a **slot** message.


```
#slot netsource 130 5370
```

After the file has been transferred the netsource sends a **finish** message to the protocol server and the protocol server replies with a **forget** message.

```
#finish 130 netsource completed 5370
```

```
#forget 130
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

A

[Abort Command message](#) 23

Abstract data model

client ([section 3.3.1](#) 43, [section 3.4.1](#) 50, [section 3.5.1](#) 53, [section 3.6.1](#) 55, [section 3.7.1](#) 83)
server ([section 3.1.1](#) 38, [section 3.2.1](#) 39)

[Applicability](#) 12

C

[Capability negotiation](#) 12

[CD Command message](#) 23

[cd message example](#) 87

[Change tracking](#) 92

Client

abstract data model ([section 3.4.1](#) 50, [section 3.5.1](#) 53)

higher-layer triggered events ([section 3.4.4](#) 51, [section 3.5.4](#) 54)

initialization ([section 3.4.3](#) 51, [section 3.5.3](#) 54)

local events ([section 3.3.7](#) 50, [section 3.6.7](#) 83, [section 3.7.7](#) 86)

message processing ([section 3.4.5](#) 51, [section 3.5.5](#) 54)

other local events ([section 3.4.7](#) 53, [section 3.5.7](#) 55)

[overview](#) 55

sequencing rules ([section 3.4.5](#) 51, [section 3.5.5](#) 54)

timer events ([section 3.4.6](#) 53, [section 3.5.6](#) 55)

timers ([section 3.4.2](#) 51, [section 3.5.2](#) 54)

Client - abstract data model ([section 3.3.1](#) 43, [section 3.6.1](#) 55, [section 3.7.1](#) 83)

Client - higher-layer triggered events
[table storage capacity shortage](#) 44

Client - initialization ([section 3.3.3](#) 43, [section 3.4.3](#) 51, [section 3.5.3](#) 54, [section 3.6.3](#) 55, [section 3.7.3](#) 84)

Client - message processing ([section 3.3.5](#) 44, [section 3.6.5](#) 55)

[create_attribute_files command](#) 84

[generic-response messages](#) 44

[jsort2.exe command](#) 84

[make_pu_diff command](#) 84

[no-response messages](#) 49

[pupdateclient command](#) 85

[receiving merge.pyc command](#) 56

[receiving pythontasks.pyc command](#) 69

[receiving reduce.pyc command](#) 71

[receiving split.pyc command](#) 82

[sp_make_pu_diff command](#) 85

[specific-response messages](#) 47

[truncate_anchorinfo command](#) 85

[truncate_clickinfo command](#) 86

Client - sequencing rules ([section 3.3.5](#) 44, [section 3.6.5](#) 55)

Client - timer events ([section 3.3.6](#) 50, [section 3.6.6](#) 83, [section 3.7.6](#) 86)

Client- sequencing rules

[create_attribute_files command](#) 84

[generic-response messages](#) 44

[jsort2.exe command](#) 84

[make_pu_diff command](#) 84

[no-response messages](#) 49

[pupdateclient command](#) 85

[receiving merge.pyc command](#) 56

[receiving pythontasks.pyc command](#) 69

[receiving reduce.pyc command](#) 71

[receiving split.pyc command](#) 82

[sp_make_pu_diff command](#) 85

[specific-response messages](#) 47

[truncate_anchorinfo command](#) 85

[truncate_clickinfo command](#) 86

[Close-stdin Command message](#) 23

[cobra commands](#) 28

Commands

[cobra](#) 28

[create_attribute_files](#) 31

[jsort2.exe](#) 31

[make_pu_diff](#) 32

[merge.pyc](#) 28

[pupdateclient](#) 32

[pythontasks.pyc](#) 28

[reduce.pyc](#) 30

[sp_make_pu_diff](#) 32

[split.pyc](#) 30

[truncate_anchorinfo](#) 33

[truncate_clickinfo](#) 33

Complex types

[dir_size tuple structure](#) 19

[directory_listing array](#) 20

[file_listing array](#) 20

[heartbeat_info structure](#) 19

[path_dir_size tuple structure](#) 20

[sortkeyspec](#) 21

[tableinputspec](#) 22

[tableoutputspec](#) 22

[target_name](#) 22

[task_info structure](#) 20

[task_info_copy structure](#) 21

[create_attribute_files command](#) 31

D

Data model - abstract

client ([section 3.3.1](#) 43, [section 3.4.1](#) 50, [section 3.5.1](#) 53, [section 3.6.1](#) 55, [section 3.7.1](#) 83)

server ([section 3.1.1](#) 38, [section 3.2.1](#) 39)

[dir_size tuple structure complex type](#) 19

[directory_listing array complex type](#) 20

[Disk_limit Command message](#) 23

E

Enumerations

[merge function](#) 16

[pythontasks function](#) 17

- [reduce functions](#) 18
- [shell commands](#) 18
- [task status](#) 16
- [task type](#) 16
- [Error Message message](#) 23
- Examples
 - [cd message](#) 87
 - [file transfer with netsink and netsource](#) 88
 - [overview](#) 87
 - [path name references](#) 87
 - [shell task](#) 88
 - [simple command message](#) 87

F

- [Fields - vendor-extensible](#) 12
- [File transfer with netsink and netsource example](#) 88
- [file_listing_array_complex_type](#) 20
- [Finish Message message](#) 24
- [Finish-tasks Command message](#) 25
- [Finish-tasks-ack Message message](#) 25
- [Forget Command message](#) 25
- Functions and parameters
 - [pythontasks](#) 29

G

- [Glossary](#) 9

H

- [Heartbeat Message message](#) 25
- [heartbeat_info_structure_complex_type](#) 19
- Higher-layer triggered events
 - client ([section 3.4.4](#) 51, [section 3.5.4](#) 54)
 - [server](#) 39
- Higher-layer triggered events - client
 - [stable storage capacity shortage](#) 44
- Higher-layer triggered events - server
 - [sending abort command](#) 41
 - [sending finish-tasks message](#) 41
 - [sending remove-files command](#) 40
 - [sending setenv command](#) 41
 - [sending shell command](#) 41
 - [starting build](#) 40
 - [starting command on worker protocol client](#) 40

I

- [Implementer - security considerations](#) 90
- [Index of security parameters](#) 90
- [Info Message message](#) 25
- [Informative references](#) 10
- Initialization
 - client ([section 3.4.3](#) 51, [section 3.5.3](#) 54)
 - server ([section 3.1.3](#) 38, [section 3.2.3](#) 40)
- Initialization - client ([section 3.3.3](#) 43, [section 3.4.3](#) 51, [section 3.5.3](#) 54, [section 3.6.3](#) 55, [section 3.7.3](#) 84)
- [Introduction](#) 9

J

- [jsort2.exe command](#) 31

L

- Local events
 - client ([section 3.3.7](#) 50, [section 3.6.7](#) 83, [section 3.7.7](#) 86)
 - [server](#) 39

M

- [make_pu_diff command](#) 32
- [merge function enumeration](#) 16
- [merge.pyc command](#) 28
- Message processing
 - client ([section 3.4.5](#) 51, [section 3.5.5](#) 54)
 - [server](#) 39
- Message processing - client ([section 3.3.5](#) 44, [section 3.6.5](#) 55)
 - [create_attribute_files command](#) 84
 - [generic-response messages](#) 44
 - [jsort2.exe command](#) 84
 - [make_pu_diff command](#) 84
 - [no-response messages](#) 49
 - [pupdateclient command](#) 85
 - [receiving merge.pyc command](#) 56
 - [receiving pythontasks.pyc command](#) 69
 - [receiving reduce.pyc command](#) 71
 - [receiving split.pyc command](#) 82
 - [sp_make_pu_diff command](#) 85
 - [specific-response messages](#) 47
 - [truncate_anchorinfo command](#) 85
 - [truncate_clickinfo command](#) 86
- Message processing - server
 - [finish-tasks-ack message](#) 41
 - [heartbeat message](#) 41
 - [info message](#) 41
 - [Netsink-Netsource messages](#) 42
 - [receiving finish message](#) 41
 - [simple_response message](#) 42
 - [stderr message](#) 42
 - [stdout message](#) 42
 - [task-info message](#) 43
- Messages
 - [Abort Command](#) 23
 - [CD Command](#) 23
 - [Close-stdin Command](#) 23
 - [cobra commands](#) 28
 - [create_attribute_files command](#) 31
 - [dir_size tuple structure complex type](#) 19
 - [directory_listing_array_complex_type](#) 20
 - [Disk_limit Command](#) 23
 - [Error Message](#) 23
 - [file_listing_array_complex_type](#) 20
 - [Finish Message](#) 24
 - [Finish-tasks Command](#) 25
 - [Finish-tasks-ack Message](#) 25
 - [Forget Command](#) 25
 - [Heartbeat Message](#) 25
 - [heartbeat_info_structure_complex_type](#) 19
 - [Info Message](#) 25
 - [jsort2.exe command](#) 31

- [make_pu_diff command](#) 32
- [merge function enumeration](#) 16
- [merge.pyc command](#) 28
- [Netsink Command](#) 25
- [Netsource Command](#) 26
- [path names simple type](#) 14
- [path_dir_size tuple structure complex type](#) 20
- [Poll-task Command](#) 26
- [pupdateclient command](#) 32
- [pythontasks function enumeration](#) 17
- [pythontasks functions and parameters](#) 29
- [pythontasks.pyc command](#) 28
- [reduce functions enumeration](#) 18
- [reduce.pyc command](#) 30
- [Remove-files Command](#) 26
- [Request_heartbeat Command](#) 27
- [Setenv Command](#) 27
- [Shell Command](#) 27
- [shell commands enumeration](#) 18
- [Simple_command Command](#) 34
- [Simple_response Message](#) 35
- [Slot Message](#) 35
- [sortkeyspec complex type](#) 21
- [sp_make_pu_diff command](#) 32
- [split.pyc command](#) 30
- [Stderr Message](#) 36
- [Stdout Message](#) 36
- [syntax](#) 13
- [tableinputspec complex type](#) 22
- [tableoutputspec complex type](#) 22
- [target_name complex type](#) 22
- [task identifiers simple type](#) 14
- [task status enumeration](#) 16
- [task type enumeration](#) 16
- [task_info structure complex type](#) 20
- [task_info_copy structure complex type](#) 21
- [Task-info Message](#) 37
- [transport](#) 13
- [truncate_anchorinfo command](#) 33
- [truncate_clickinfo command](#) 33

N

- [Netsink Command message](#) 25
- [Netsource Command message](#) 26
- [Normative references](#) 10

O

- Other local events
 - client ([section 3.4.7](#) 53, [section 3.5.7](#) 55)
 - server 43
- [Overview \(synopsis\)](#) 11

P

- [Parameters - security index](#) 90
- [Path name references example](#) 87
- [Path names simple type](#) 14
- [path_dir_size tuple structure complex type](#) 20
- [Poll-task Command message](#) 26
- [Preconditions](#) 12

- [Prerequisites](#) 12
- [Product behavior](#) 91
- [pupdateclient command](#) 32
- [pythontasks function enumeration](#) 17
- [pythontasks functions and parameters](#) 29
- [pythontasks.pyc command](#) 28

R

- [reduce function enumeration](#) 18
- [reduce.pyc command](#) 30
- References
 - [informative](#) 10
 - [normative](#) 10
- [Relationship to other protocols](#) 11
- [Remove-files Command message](#) 26
- [Request_heartbeat Command message](#) 27

S

- Security
 - [implementer considerations](#) 90
 - [parameter index](#) 90
- Sequencing rules
 - client ([section 3.4.5](#) 51, [section 3.5.5](#) 54)
 - server 39
- Sequencing rules - client ([section 3.3.5](#) 44, [section 3.6.5](#) 55)
 - [create_attribute_files command](#) 84
 - [generic-response messages](#) 44
 - [jsort2.exe command](#) 84
 - [make_pu_diff command](#) 84
 - [no-response messages](#) 49
 - [pupdateclient command](#) 85
 - [receiving merge.pyc command](#) 56
 - [receiving pythontasks.pyc command](#) 69
 - [receiving reduce.pyc command](#) 71
 - [receiving split.pyc command](#) 82
 - [Sp_make_pu_diff Command](#) 85
 - [specific-response messages](#) 47
 - [truncate_anchorinfo command](#) 85
 - [truncate_clickinfo command](#) 86
- Sequencing rules - server
 - [finish-tasks-ack message](#) 41
 - [heartbeat message](#) 41
 - [info message](#) 41
 - [Netsink-Netsource messages](#) 42
 - [receiving finish message](#) 41
 - [simple_response message](#) 42
 - [stderr message](#) 42
 - [stdout message](#) 42
 - [task-info message](#) 43
- Server
 - abstract data model ([section 3.1.1](#) 38, [section 3.2.1](#) 39)
 - [higher-layer triggered events](#) 39
 - initialization ([section 3.1.3](#) 38, [section 3.2.3](#) 40)
 - [local events](#) 39
 - [message processing](#) 39
 - [other local events](#) 43
 - [sequencing rules](#) 39
 - [timer events](#) 39

- [timers](#) 38
- Server - higher-layer triggered events
 - [sending abort command](#) 41
 - [sending finish-tasks message](#) 41
 - [sending remove-files command](#) 40
 - [sending setenv command](#) 41
 - [Sending Shell Command](#) 41
 - [starting build](#) 40
 - [starting command on worker protocol client](#) 40
- Server - message processing
 - [finish-tasks-ack message](#) 41
 - [heartbeat message](#) 41
 - [info message](#) 41
 - [Netsink-Netsource messages](#) 42
 - [receiving finish message](#) 41
 - [simple_response message](#) 42
 - [stderr message](#) 42
 - [stdout message](#) 42
 - [task-info message](#) 43
- Server - sequencing rules
 - [finish-tasks-ack message](#) 41
 - [heartbeat message](#) 41
 - [info message](#) 41
 - [Netsink-Netsource messages](#) 42
 - [receiving finish message](#) 41
 - [simple_response message](#) 42
 - [stderr message](#) 42
 - [stdout message](#) 42
 - [task-info message](#) 43
- Server - timer events
 - [Heartbeat Timeout](#) 43
- Server - timers
 - [build timeout timer](#) 39
 - [heartbeat timeout timer](#) 39
 - [heartbeat timer](#) 39
- [Setenv Command message](#) 27
- [Shell Command message](#) 27
- [shell commands enumeration](#) 18
- [shell_task example](#) 88
- Simple types
 - [path names](#) 14
 - [task identifiers](#) 14
 - [transfer key](#) 15
- [Simple_command Command message](#) 34
- [simple_command message example](#) 87
- [Simple_response Message message](#) 35
- [Slot Message message](#) 35
- [sortkeyspec complex type](#) 21
- [sp_make_pu_diff command](#) 32
- [split.pyc command](#) 30
- [Standards assignments](#) 12
- [Stderr Message message](#) 36
- [Stdout Message message](#) 36
- Syntax
 - [messages - overview](#) 13

T

- [tableinputspec complex type](#) 22
- [tableoutputspec complex type](#) 22
- [target_name complex type](#) 22
- [Task identifiers simple type](#) 14

- [task status enumeration](#) 16
- task type
 - [enumeration](#) 16
- [task type enumeration](#) 16
- [task_info structure complex type](#) 20
- [task_info_copy structure complex type](#) 21
- [Task-info Message message](#) 37
- Timer events
 - client ([section 3.4.6](#) 53, [section 3.5.6](#) 55)
 - [server](#) 39
- Timer events - client ([section 3.3.6](#) 50, [section 3.6.6](#) 83, [section 3.7.6](#) 86)
- Timer events - server
 - [heartbeat timeout](#) 43
- Timers
 - client ([section 3.4.2](#) 51, [section 3.5.2](#) 54)
 - [server](#) 38
- Timers - server
 - [build timeout timer](#) 39
 - [heartbeat timeout timer](#) 39
 - [heartbeat timer](#) 39
- [Tracking changes](#) 92
- [Transfer key simple type](#) 15
- [Transport](#) 13
- Triggered events - client
 - [stable storage capacity shortage](#) 44
- Triggered events - higher layer
 - [server](#) 39
- Triggered events - higher-layer
 - client ([section 3.4.4](#) 51, [section 3.5.4](#) 54)
- Triggered events - server
 - [sending abort command](#) 41
 - [sending finish-tasks message](#) 41
 - [sending remove-files command](#) 40
 - [sending setenv command](#) 41
 - [Sending Shell Command](#) 41
 - [starting build](#) 40
 - [starting command on worker protocol client](#) 40
- [truncate_anchorinfo command](#) 33
- [truncate_clickinfo command](#) 33

V

- [Vendor-extensible fields](#) 12
- [Versioning](#) 12